Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

MECCANICA E SCIENZE AVANZATE DELL'INGEGNERIA

Ciclo 35

Settore Concorsuale: 09/A2 - MECCANICA APPLICATA ALLE MACCHINE

Settore Scientifico Disciplinare: ING-IND/13 - MECCANICA APPLICATA ALLE MACCHINE

A ROS-BASED SOFTWARE ARCHITECTURE FOR A VERSATILE COLLABORATIVE DUAL-ARMED AUTONOMOUS MOBILE ROBOT FOR THE MANUFACTURING INDUSTRY

Presentata da: Simone Comari

Coordinatore Dottorato	**Supervisore**
Lorenzo Donati	Marco Carricato

Esame finale anno 2023

Abstract

The current global industrial context is undergoing a profound paradigm shift whose event can be traced back to the incessant technological development fueled by the Internet, new design methods, and, generally speaking, the innumerable possibilities granted by Information Technology. The fourth industrial revolution, more commonly addressed as *Industry 4.0*, counts integration, flexibility and optimisation as its fundamental pillars, and, in this context, *Human-Robot Collaboration* has become a key driver for manufacturing sustainability in Europe. Low installation and running costs and a high degree of flexibility are the main characteristics that make *collaborative robots (or cobots)* appealing to many companies seeking out re-shoring production facilities with a short return on investment. Cobots, in fact, differently from traditional industrial robots, can be installed in a fenceless but completely safe shared environment thanks to their intrinsic safety-related features. Despite the many examples of collaborative robotics applied in industrial scenarios, there still exists a gap between the promising results shown by the academic community and the availability of industry-proof products on the market. One of the main concerns of the industry stakeholders is related to safety when introducing a cobot in a shared working area.

The *ROSSINI European project*, especially thanks to a novel safety-camera system, aims to implement a true Human-Robot Collaboration, laying the ground for a new working paradigm at the industrial level. The main goal of the ROSSINI project is to design, develop and demonstrate a modular and scalable platform for the integration of human-centred robotic technologies in industrial production environments. To prove the versatility of the proposed framework, three industrial use cases were selected to deploy the new technology. The need for a software architecture suitable to the robotic platform employed in one of these use cases was the main trigger of this Thesis. The application consists in the automatic loading and unloading of raw-material reels to an automatic packaging machine through an *Autonomous Mobile Robot* composed by an Autonomous Guided Vehicle, two collaborative manipulators, and an eye-on-hand vision system for performing tasks in a partially unstructured environment.

The results obtained during the ROSSINI use case development were later used in a spin-off project, namely the *SENECA project*. The motivation for this second work arises from a very specific industrial need in the pharmaceutical world that is still not fully addressed at the market level: the robot-driven automatic cleaning of pharmaceutical bins. The proposed solution includes the implementation of a procedure for automatic scanning path generation given a CAD model of a pharmaceutical bin that includes collision avoidance and kinematic constraints, and the realization of a Deep Learning-based binary classifier for surface inspection.

The substantial difference of the industrial context in which the same Autonomous Mobile Robot was deployed with little hardware and software adjustments is evidence of the inherent versatility of mobile collaborative robots, a proof of their positive impact on diverse production lines, and a motivation for future investments in the research about Human-Robot Collaboration by the Industry.

Acronyms

ACO Ant Colony Optimization. 60, 105, 110, 126

AGV Autonomous Guided Vehicle. 1, 28, 30, 38–40, 42, 43, 45, 47–50, 54, 58, 59, 62, 63, 88, 89, 91

AI Artificial Intelligence. 113, 114, 156

AMR Autonomous Mobile Robot. 1, 37, 38, 42–46, 48, 49, 52–55, 58, 59, 61–63, 67, 68, 79, 87, 90, 123, 124, 155

API Application Programming Interface. 54, 88

CAD Computer-Aided Design. 1, 24, 34, 35, 39, 50, 51, 55, 60, 67, 68, 82, 96, 98, 106, 107, 124, 155

CCD Charged Coupled Device. 21, 22

CMOS Complementary Metal-Oxide Semiconductor. 22

CNN Convolutional Neural Network. 95, 114–119, 124, 126, 156

CV Computer Vision. 7, 12, 24–26, 114, 131, 144, 155

DL Deep Learning. 1, 7, 12, 24–26, 35, 113, 114, 119, 125, 153, 155

DLT Direct Linear Transformation. 138

DNN Deep Neural Network. 114–116, 120, 150–152, 157

DoV Distance of View. 108, 109

EU European Union. 7, 16, 26, 27, 29, 61, 88, 123, 124

EuRoC European programme on Robotics Challenges. 7, 28, 30, 123

FCL Flexible Collision Library. 57

FPGA Field Programmable Gate Array. 23

GA Greedy Area. 102–104, 108–110, 156

GPU Graphics Processing Unit. 23, 25, 27, 47, 53, 124, 152, 153

HG Hand Guiding. 18

HR Human-Robot. 27

HRC Human-Robot Collaboration. 1, 7, 13, 15, 26, 27, 34, 47, 124

HSV Hue Saturation Value. 115

HTTP HyperText Transfer Protocol. 54, 55, 59

ICT Information and Communication Technologies. 11

IEC International Electrotechnical Commission. 16

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 116, 117, 156

IMA Industria Macchine Automatiche. 28, 29, 32–34, 37, 41, 61, 62, 123, 155, 159

IoT Internet of Things. 11, 12

IP Internet Protocol. 43, 55

IPC Industrial Personal Computer. 45, 47, 49, 123

ISO International Organisation for Standardisation. 13, 16, 17, 19

IT Information Technology. 1, 11, 54

KPI Key Performance Index. 124

LAN Local Area Network. 42, 45, 47, 49

LED Light-Emitting Diode. 23

MAC Metropolis Acceptance Criterion. 104, 105

MaXima Multiple Actions for Innovation in Machine Automation. 7, 13, 30, 31, 37, 39–43, 45–47, 54, 61, 66, 123

MISE Ministero dello Sviluppo Economico. 30

ML Machine Learning. 12, 27, 113, 119

MOSFET Metal-Oxide Semiconductor Field-Effect Transistors. 22

NN Neural Network. 25, 66, 113, 120, 149–153

OMPL Open Motion Planning Library. 56, 57

OSRF Open Source Robotics Foundation. 139

PFL Power and Force Limiting. 19

PL Performance Level. 17

PLC Programmable Logic Controller. 45, 47, 48

PLr Performance Level rating. 17

R&I Research and Innovation. 28, 29

ReLU Rectified Linear Unit. 150

REST REpresentational State Transfer. 54, 55

RGB Red Green Blue. 23, 52, 112, 115, 118

RGB-D Red Green Blue - Depth. 23

RMS Resilient Manufacturing System. 31

ROS Robotic Operating System. 9, 34, 49, 52–55, 57–61, 88, 89, 111, 119, 123–125, 139–141, 157

ROSSINI RObot enhanced SenSing, INtelligence and actuation to Improve job quality in manufacturing. 1, 7, 8, 12, 13, 26–31, 34, 37, 45–53, 55, 57–59, 61–64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86–88, 90–92, 95–97, 123, 124, 155

RPC Remote Procedure Call. 54, 141

RRT Rapidly-exploring Random Tree. 57

RS^4 ROSSINI Smart and Safe Sensing System. 27, 34, 47–49, 58, 60, 62, 63, 124

RSACA ROSSINI Safety-Aware Control Architecture. 27

SA Simulated Annealing. 102, 103, 106, 108–110, 156, 159

SDK Software Development Kit. 52, 139, 140

SENECA Systems Enabling Efficient Cognitive Automation. 1, 7, 8, 12, 13, 31, 33, 34, 37, 45, 47, 49–53, 55, 57, 59, 60, 95, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118–120, 122, 124, 125

SGD Stochastic Gradient Descent. 152, 153

SMS Safety-rated Monitored Stop. 18, 30, 48, 49, 58, 62, 88, 123

SSM Speed and Separation Monitoring. 18, 48, 62, 88

TCP Transmission Control Protocol. 43, 141

TCP Tool-Centre Point. 57, 67, 79, 81, 89

TCP/IP Transmission Control Protocol/Internet Protocol. 43, 54

TGA TanGent Angle. 144–148, 157, 161

ToF Time-of-Flight. 23

TRL Technology Readiness Level. 123

TS Technical Specification. 19

VP ViewPoint. 98

XML eXtensible Markup Language. 54

XML-RPC XML-Remote Procedure Call. 54, 55, 58

Contents

Chapter 1

Introduction

"Robotnik", this is the term that marks the origin of the name *robot*, first used by the Czechoslovak Karel Capek in a play in 1920. Literally translating, the meaning of the word is "servant" or "forced labourer"; in fact, the birth of such programmable machines, either autonomous or semi-autonomous, arose precisely from the desire to dispense man from tiring, repetitive or dangerous work. Nonetheless, it was not until the 1970s that the first industrial robot made its debut on an assembly line: the ASEA IRb 6, an anthropomorphic robot that replicated a human arm and was used for polishing steel pipes.

These were the years of the so-called third industrial revolution, or *Industry 3.0*. Occurring in the last decades of the 20th century, it started with the development of Information Technology (IT) systems and computers as part of the digital era and was characterised by an increase in the automation and speed of processes thanks to Information and Communication Technologies (ICT) and electronic technologies.

The current global industrial context is, however, undergoing a profound paradigm shift whose event can be traced back to the incessant technological development fuelled by the Internet, new design methods, and, generally speaking, the innumerable possibilities granted by IT. The scenario we are facing now, although constantly evolving, sees as its ultimate goal the integration of industrial plants with the Internet of Things (IoT); the aim is to guarantee stable, constant communication and remote access between devices thanks to the Net [1]. In substance, we are describing the fourth industrial revolution, more commonly addressed as *Industry 4.0*, which counts integration, flexibility and optimisation as its fundamental pillars [2].

The historical scope of the fourth industrial revolution can only be compared to that of the first one (Fig. 1.1) because it radically changes the way products are originally conceived and designed. Whereas the third industrial revolution led to a vertical development of systems, improving each process autonomously, the focus of the Industry 4.0 is horizontal. The aim is to increase the synergy and interconnection between all the processes involved. In practice, we are moving from the mass market to the mass hyper-customization of products generated by industrial processes in which machines communicate with each other through digital tools that enable hyper-automation.

The basic characteristic of 4.0 is *inclusiveness*: the new 4.0 technologies are generic by conception, so by their intrinsic nature, they affect every type of enterprise, from start-ups to multinationals, and every sector, from services to manufacturing.

The technological and economic transformation of Industry 4.0 proceeds along

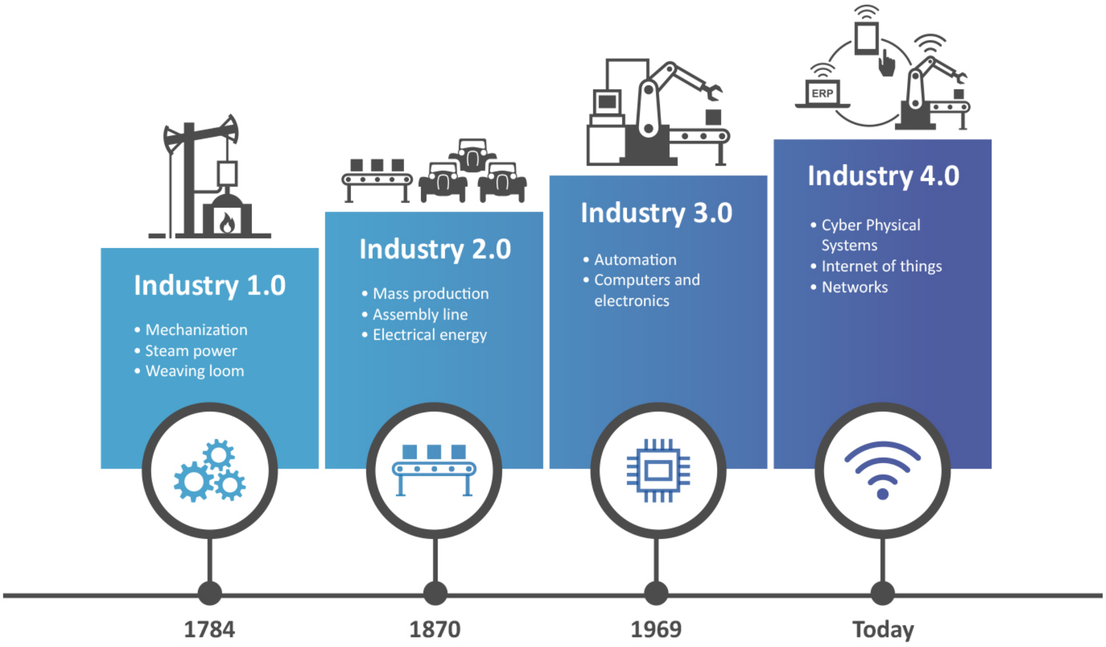


Figure 1.1: Industry revolutions. Source: https://sii.pl/blog/en/industry-4-0-the-industrial-revolution.

four parallel paths:

- *data usage*: technologies for centralising and storing data are big data, open data, IoT, machine-to-machine and cloud computing;
- *analytics*: Machine Learning, a technology that "learns" from the collected and analysed data, which allows finding hidden patterns in data sets and extracting new knowledge;
- *human-computer interaction*: augmented reality and touch interfaces;
- *digital-to-real transition*: technologies that aim to create a communication bridge between digital and real, such as additive manufacturing, 3D printing, robotics, communications, digital twin and machine-to-machine interactions.

The origin of this Thesis is to be found in the contribution to two innovative industrial projects that are framed in the context of Industry 4.0, ROSSINI and SENECA. The Thesis is structured as follows:

1. **Introduction**: a quick introduction about collaborative robots, with a short retrospective on industrial robots, opens the chapter; it is followed by some details about the norms, laws and standards that are applied when dealing with human-robot interactions, and with a few successful examples of mobile collaborative robotics application in the industry field; Sec. 1.2 is dedicated to machine vision, with a further distinction between 2D and 3D sensors and a brief discussion about advantages and drawbacks of using traditional Computer Vision against Deep Learning for image processing and analysis; Sec. 1.3 describes the history, motivations and objectives of the ROSSINI European project, which is the main drive of this work; similarly, Sec. 1.4 provides an overview of the SENECA project, which acted as a spin-off of ROSSINI, with which it shares some key elements;

2. **Experimental setup**: the experimental setup from both hardware and software perspective is illustrated in this chapter; Sec. 2.1 is dedicated to the precursor of the use case targeted by the ROSSINI project, the MaXima project, with which it shares the work cell organization and main objective; in Sec. 2.2, the hardware setup of both ROSSINI and SENECA projects is described, starting from the common elements up to their own peculiarities; along the same lines, Sec. 2.3 defines the software setup of both projects;

3. **ROSSINI use case**: a detailed description of the ROSSINI use case is given in this chapter, beginning with calibration, moving to the details of each step in the operation, and ending with the definition of fall-back procedures; due to the difference in the objectives of ROSSINI and SENECA, project-related results and discussions are placed at the end of the chapter dedicated to each project;

4. **SENECA use case**: the three main components of this use case are described in this chapter; given their different domains, each one of the three sections follows the same pattern: methodology, implementation and results, if applicable;

5. **Conclusions**: conclusions are drawn, and future works are outlined.

1.1 Collaborative robots

In the context of Industry 4.0, *Human-Robot Collaboration (HRC)* has become a key driver for manufacturing sustainability in Europe. Low installation and running costs and a high degree of flexibility are the main characteristics that make collaborative robots (or cobots[1]) appealing to many companies seeking out re-shoring production facilities with a short return on investment [3], [4].

Moreover, the ageing of the workforce poses a dramatic challenge to health and safety in all European countries [5]. In this context, cobots can become an important support and an extension of human capabilities, thus relieving operators from repetitive, alienating and/or heavy operations and leaving them in charge of more complex tasks, where experience can assume predominant importance and greater efficacy.

Cobots, in fact, differently from traditional industrial robots, can be installed in a fenceless but completely safe shared environment, thus leaving the factory's layout and facilities almost unaffected. This is thanks to their intrinsic safety-related features obtained by the integration of suitable sensors, safety-rated computing units and communication channels, and certified control software.

1.1.1 Industrial robots

Before digging into the specific features of collaborative robots, it may be useful to briefly look into the technology that pioneered them during the third industrial revolution: industrial robotics.

Quoting from ISO 8373:2012 [6], an industrial robot is an "automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications". A

[1] The term "cobot" was coined in 1996 by J. E. Colgate and M. Peshkin, professors at Northwestern University, who issued a homonym US patent in 1997 to describe this new technology.

robot is designed to move, monitor or inspect materials, parts or tools through variable motions programmed according to each specific task. It acquires information from the environment and acts in response. In a more visionary definition, a robot is an intelligent connection between perception and action.

A robot is, therefore, a complex machine composed by:

- a *mechanical system* for interacting with the environment and providing structure;
- an *actuation system* that allows motion and task execution;
- a *sensory system* that provides feedback and acquires information about both environment (*exteroceptive sensors*) and the internal status (*proprioceptive sensors*);
- a *control system* for the run-time control and programming.

Due to the many elements involved, robotics is an interdisciplinary science with competencies from the fields of mechanics, electronics, computer science, automation, material science and sensor technologies.

According to the combination and configuration of these components, industrial robots can be suitably designed to achieve various payloads, maximum reach, dexterity (often linked to the number of axes), rigidity, repeatability, weight and speed. Reference ranges for some of these indicators are reported in Table 1.1.

Indicator	**Range**
Number of axes	3-6
Payload	3-200 [kg]
Maximum reach	500-3000 [mm]
Maximum joint speed	600 [deg/s]
Repeatability	0.01-0.1 [mm]
Weight	30-1500 [kg]

Table 1.1: Ranges of most relevant indicators for an industrial robot.

It is evident how the inherent versatility of such complex systems can suit a large set of applications and fields and be declined in several ways, such as:

- manufacturing industry;
- entertainment (toys, movie models, etc.);
- service (cleaning, mopping, mowing, etc.);
- legged robotics (e.g. exploration on rough/dangerous terrains);
- space (e.g. maintenance, recovery, exploration);
- micro/nanorobotics;
- exoskeletons/power amplifiers;
- underwater robotics (e.g. exploration, data collection, maintenance);
- medicine (prosthetic, rehabilitation, surgery);
- social sciences (interaction with humans, study of human behaviour);
- agriculture (harvest, seeding, irrigation, etc.);

- construction (transportation, manipulation, inspection);
- education.

Given the scope of this Thesis, it is worth mentioning some specific applications to the manufacturing field that greatly benefit from the employment of industrial robots:

- pick and place;
- (de)palletization;
- machine tending;
- selection and sorting;
- packaging;
- quality control and inspection;
- sanity check;
- welding;
- spray painting and coating;
- bonding and sealing;
- laser/water cutting;
- deburring, polishing and grinding;
- tightening, wiring and fixing;
- assembly.

1.1.2 Safety standards in Human-Robot Collaboration

The widespread use of robots in the industry has made it necessary to carefully examine issues related to the safety of working environments, especially in recent years when human operators are progressively in close contact with robots [7].

Before 2006, regulations in Europe required a clear separation of the environments in which robots operated from those in which human operators worked because most commercial products did not allow for safe cooperation between humans and robots. In order to ensure the maximum safety of people while performing their tasks, physical or optical barriers were erected to separate the spaces in which the robots operated from the areas accessible to the operators. Using different types of sensors to monitor intrusions or malfunctions was usually necessary so that the robots could be fully stopped if necessary. Sometimes, these restrictions caused delays in industrial production compared to the level of efficiency that could have been achieved through safe collaboration between humans and robots or simply if operators and robots could have shared the same space without barriers of any kind.

The definition of recent standards and the availability of equipment capable of ensuring increasingly safe cooperation has gradually made it possible to limit or remove physical barriers provided certain conditions are met. These conditions are essentially related to the need to monitor the position and movements of operators appropriately and, at the same time, to have robots equipped with technology that allows them to promptly react when hazardous situations occur. As introduced in Sec. 1.1, robots that are purposely designed to interact with humans and share a barrier-free space with them are called "collaborative"; those that are not are commonly called "industrial". Although there seems to be a clear division between industrial and collaborative robots, with the addition of special sensors and the necessary precautions, it is possible to reduce or remove the physical barriers that delimit even an industrial robot. In the latter case, however, the performance of an industrial robot must be decreased typically by a reduction in speed or by the execution of longer trajectories.

To better understand the regulatory aspects of robot use, Sec. 1.1.2.1 introduces the main robot safety regulations.

1.1.2.1 Norms, laws, and standards in robotics

European Union (EU) directives are European legislative instruments that guarantee the protection of people's health in the workplace and possibly also that of the environment. EU directives establish the essential safety requirements, while *technical standards* indicate the recommended technical solutions for achieving them. Technical standards are drafted at the international level by standardisation institutions operating at global, European or national levels. These bodies are, globally, the International Organisation for Standardisation (ISO) for the mechanical engineering sector and the International Electrotechnical Commission (IEC) for the electrical sector.

Machinery safety standards are divided into three types:

1. **type A standards** (basic standards): they contain the basic concepts, design principles and general aspects applicable to all machines;
2. **type B standards** (group standards): they deal with a specific aspect of safety or a safety device, and they are, in turn, divided into two groups:
   (a) **type B1**: they deal with safety details;
   (b) **type B2**: they deal with protective devices;
3. **type C standards** (machine family standards): they deal with safety requirements per type of machine.

Fig. 1.2 shows the main regulations and directives related to the safety of the people working in environments where robots may carry out collaborative operations. Before placing a new machine on the market, manufacturers must ensure that the machine complies with all related directives and standards, make a technical dossier available in the event of a justified request by an authority, sign a "Declaration of Conformity", and affix the CE marking. Since the intended use of a robot is not known in advance, it cannot be CE marked. In fact, a robot is considered a partial machine and, therefore, can only have the declaration of incorporation. The task of the installer/integrator is to carry out the risk analysis and produce the user manual and technical file before placing the CE mark.

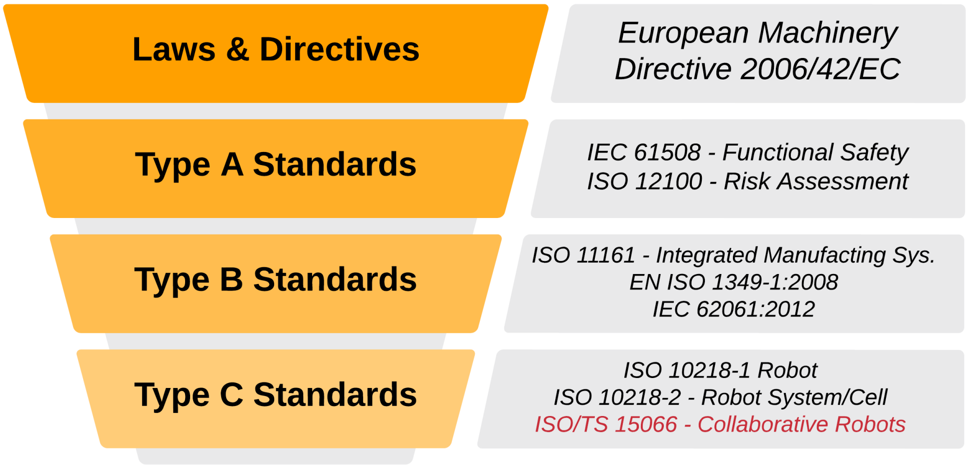

Laws & Directives
European Machinery Directive 2006/42/EC
Type A Standards
IEC 61508 - Functional Safety
ISO 12100 - Risk Assessment
Type B Standards
ISO 11161 - Integrated Manufacting Sys.
EN ISO 1349-1:2008
IEC 62061:2012
Type C Standards
ISO 10218-1 Robot
ISO 10218-2 - Robot System/Cell
ISO/TS 15066 - Collaborative Robots

Figure 1.2: Main standards and directive for the use of robots in industry.

Generically speaking, the risk assessment consists of the identification, evaluation and estimation of the levels of risk involved in a situation, the comparison of the risks against benchmarks or standards and the determination of an acceptable level of risk. In the robotic field, a risk assessment is used to evaluate potential hazards that may be harmful to a human worker during the operation of a robotic system and to mitigate them to acceptable levels by an iterative process.

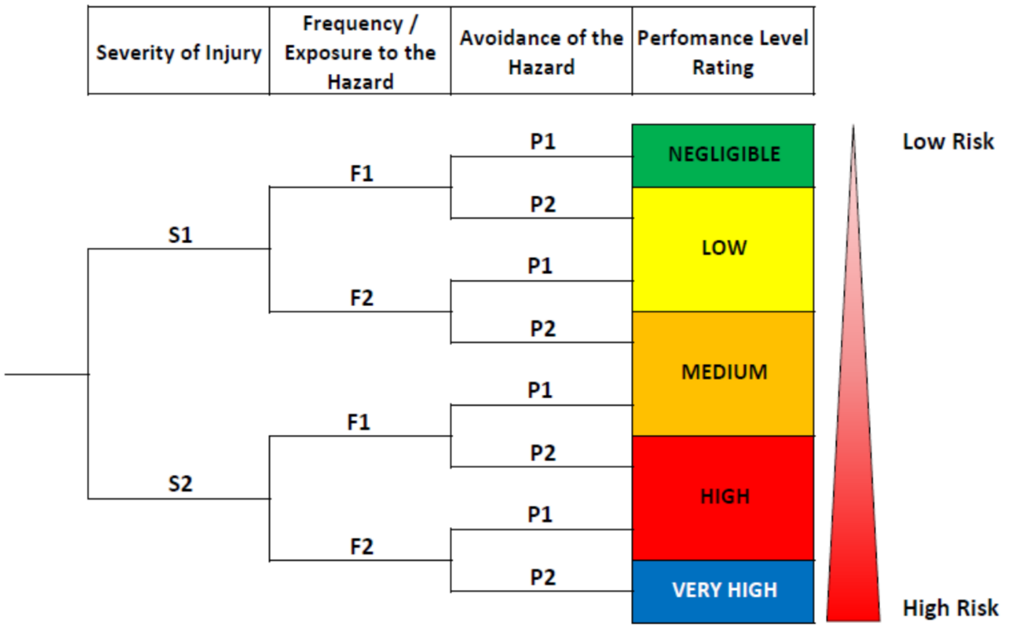


Figure 1.3: Decision tree for the risk evaluation (PLr).

With reference to Fig. 1.3, the risk is quantified using the *Performance Level rating (PLr)*, a 5-element scale from NEGLIGIBLE to VERY HIGH that depends on three factors:

1. **severity of injury (S)**: it may be slight (S1), i.e. reversible injury, or serious (S2), i.e. irreversible injury or death;

2. **frequency and/or exposure to hazard (F)**: it divides into seldom and/or short exposure time (F1) and frequent and/or long exposure time (F2);

3. **possibility of avoiding hazard or limiting harm (P)**: it may be possible under specific condition (P1) or scarcely possible (P2).

ISO 13849-1 [8] defines the concept of *Performance Level (PL)* as the probability of residual risk of a component or machine. PLs are classified into five levels, from "a" to "e" as the risk increases; each level identifies a numerical range of average probability of dangerous failure per hour, as shown in Table 1.2. Noticeably, there is a one-on-one correspondence between the PLr and the PL. For instance, if the cell or application has an estimated PLr = HIGH, the integrator/user must ensure that the safety features that secure that application have a PL ≥ d.

Going into more detail about the specific standard for robots (i.e. type C), ISO 10218-1/2 [9]-[10] introduce the concept of collaborative operations. Under these circumstances, the human operators are allowed to enter a safeguarded space and/or interact with the robot as long as a few protection measurements are present and the following constraints are met:

PLr	PL	Average probability of dangerous failure per hour
NEGLIGIBLE	a	$\geq 10^{-5} \wedge < 10^{-4}$
LOW	b	$\geq 3 \cdot 10^{-6} \wedge < 10^{-5}$
MEDIUM	c	$\geq 10^{-6} \wedge < 3 \cdot 10^{-6}$
HIGH	d	$\geq 10^{-7} \wedge < 10^{-6}$
VERY HIGH	e	$\geq 10^{-8} \wedge < 10^{-7}$

Table 1.2: PL values and correspondence between PL and PLr. $\wedge$ represents the logical *and*.

- the operator must have complete control of the robot;
- in case of close proximity with the robot, sensors or other means must limit (or stop) the speed and wrenches of the robot.

For the sake of completeness, we report in the following the traditional operation usually adopted for industrial robots:

- only specialised operators may enter the cell;
- the automatic operation mode cannot be activated (or remain in operation);
- the maximum permitted robot speed is 250 mm/s (this applies to any point belonging to the robot);
- the operator must have a programming button panel;
- no external commands may be activated (with the exception of the emergency stop).

There are four types of collaborative operations:

1. **Safety-rated Monitored Stop (SMS)**: the robot stops when the operator enters the collaborative workspace and resumes when the operator leaves that space; to ensure this mode of operation, robots can be monitored with laser scanners or vision systems that detect the presence of operators; the robot, in this case, is not switched off, but its motors are braked, and its movements are monitored;
2. **Hand Guiding (HG)**: the robot's movements are controlled directly by the operator, who guides it manually; for this mode, the robot must have a device that allows it to sense external forces on its tool, such as a torque sensor or a safety-rated controller that can estimate them;
3. **Speed and Separation Monitoring (SSM)**: contact between the operator and moving robot is prevented by the latter, i.e. the robot always maintains a safe distance from the operator; this type of cooperation is achieved by delimiting different safety zones in the robot's working area; certain zones will allow the maximum speed of the robot, while other zones will require lower speeds or even stop the robot, depending on the proximity of the operators and the level of risk of the ongoing operation; monitoring of the various zones is performed by different equipment, mainly vision systems, and each zone can have any shape and size;

4. **Power and Force Limiting (PFL)**: contact forces between operator and robot are limited by a certain safety threshold; with proper design and control, the robot may be able to sense external forces applied to it; thus, in the event of dangerous contact, the robot imparts limited static and dynamic forces; in other words, once the robot is hit (i.e. touched, unintentionally, by an operator), the brakes and actuators act to provide less energy in the direction of impact; some robots may simply stop, while others may move in the opposite direction of impact according to the particular design.

An addendum to the ISO 10218-1/2 is given by the ISO/TS 15066 [11], which is not a standard, but a Technical Specification (TS). Its scope is to better explain the four collaborative operations, which requisites must be satisfied, and how to assess the risks. For instance, it includes a study about the pain threshold for different parts of the body in relation to forces or pressures that can be used to evaluate the risk associated with a particular application and identify criticalities. Another example is given by a section dedicated to the instruction about how to dynamically calculate the safety distances so as to reduce the safety area surrounding a moving robot in a safety-monitored scenario under some specific conditions.

Lastly, it should be noted that using a collaborative robot does not necessarily mean that the application is collaborative and vice versa. In fact, for some collaborative operations, industrial robots can be used if there are external safety sensors that meet all the requirements. At the same time, a collaborative robot that, for instance, manipulates a sharp object automatically may lose all the benefits of being inherently safe.

1.1.3 Mobile cobots in the manufacturing industry

While industrial robotics has been heavily employed in the past decades, collaborative robotics, despite being an available technology since the early 2000s, prominently entered the industrial scene just recently [12]. Given their versatility and inherent characteristic of not needing any protective barrier surrounding them, it was not long before researchers realized it was a good idea to install them on mobile platforms to freely move them around to perform different jobs. Moreover, a single mobile robot can replace many stationary robots that would otherwise only be operational for a short period of time, thus considerably reducing the fixed costs of a production line. Among the first examples of integrated mobile robots and robotic manipulators for industry-oriented tasks, in [13] and [14], the authors designed and developed original robotic systems able to navigate and operate in a specific environment (e.g. a grocery store [13]) and perform a complex task (e.g. bin picking [14]). The Robo-Partner EU project [15] focused on combining the capacities and cognitive abilities of humans with the robot strength, velocity, repeatability and precision in the assembly of the rear axle of a passenger vehicle, developing intuitive human-robot interfaces using sensors, visual servoings, speech recognition, and advanced control algorithms. [16] addressed the automatization of order-picking procedures, investigating the use case of autonomous picking and palletizing by means of a dedicated mobile platform equipped with an industrial cobot. The VALERI EU project [17] aimed at testing mobile robotics in the aerospace industry. The Kuka's omniRob platform was used and supplemented with a rotating vertical linear axis on which a lightweight manipulator was mounted. Tactile sensors and a 2.5D vision system were employed to establish a safe space around the tool in order to robustly detect humans or unknown objects intruding into this safety area. In [18], a similar robotic system to the one employed in this work was used and equipped with multiple perceptive sensors to perform manipulation tasks in a water pump production site. In [19], the authors presented a mobile robotic system designed for operating on the moving floors of automotive final assembly lines. A mobile manipulation system for automated logistic applications was presented in [20] and [21], where the authors changed the picking strategy switching from lifting to dragging items on board, with the main benefit of reduced payload requirements on the robotic arm. A 3D in-hand vision system was used to detect parcel boxes on a pallet, making this system able to cope with the common imperfections in the items' alignment in real industrial scenarios. A different navigation and planning paradigm is described in [22], where deep reinforcement learning was selected to perform collision-free trajectories for pick-and-place operations in highly unstructured areas, such as a logistic facility. Lastly, a comprehensive review of system architectures and applications in the field of collaborative mobile industrial manipulators is given in [23] whereas a most recent work by Bi et al. [24] discusses the main challenges and methods related to safety assurance when dealing with cobots by listing relevant examples and successful case studies.

1.2 Machine vision

Machine vision is the science that lets industrial automation systems perceive the external world through the elaboration of images, similar to the human sense of vision [25]. Hence, machine vision starts with the acquisition of pictures via cameras. Captured frames are consecutively manipulated and analysed through image processing algorithms to finally obtain relevant information and characteristics of either a product or the external environment where the vision system is deployed. Thanks to this visual feedback, the associated machine (or robot) is then able to properly react, stop, execute a particular action or simply convey the resulting information to other agents.

The high interest that the industry has been showing in this technology is mainly related to the positive cost-benefit ratio. Industrial cameras (Fig. 1.4), in fact, are able to guarantee higher levels of reliability and efficiency than traditional solutions for a relatively low cost while reducing time-to-market thanks to their inherent flexibility. Often a reprogramming or refactoring of the image processing software is the only step needed to make the system suitable for a different task or for the deployment on a new plant. The power of this technology lies in the large and diverse amount of information that can be obtained from images or video streams. On the other hand, extracting valuable data from a raw frame can be complex, computationally expensive and time-consuming. Effective, efficient and flexible methods to cope with these drawbacks have been studied and developed for decades, and accessible solutions are readily available today. Starting from smart cameras with integrated processing units to highly optimized software for image analysis and manipulation, the number of products which implement the most common computer-vision algorithms in plug-and-play format is constantly growing, thus making this powerful technology appealing to most industry fields.

The industrial fields of application mainly concerned with machine vision are:

- **position detection**: specific objects are detected, and their presence or their coordinates (position and orientation) are made available;

Figure 1.4: Different industrial camera models. Source: SVS-Vistek GmbH.

- **inspection**: using image analysis, the quality of the product, the completeness of parts of an assembly or the presence of defects are verified;
- **measurement**: the characteristics of an object are acquired in one or more of three dimensions (length, height, depth, area or volume);
- **identification**: labels are read and decoded for identifying and tracking products, regardless of the type of 1D or 2D code (e.g. bar-code, QR-code) used and its orientation; most recent works may be able to achieve the same goals without labels too.

Depending on the application, one can choose between different image acquisition and illumination technologies to highlight the salient features of the analysed object. The main distinction between types of machine vision sees two candidates: 2D vs 3D vision systems.

Disregarding the core technology, camera sensors are typically associated with a controller running proprietary software, which includes many functions from object recognition to filtering. The controller can be either on a dedicated computing unit or on a general-purpose workstation where the necessary software and drivers can be installed and deployed. In the past decade, the advancement of technologies in the field of microprocessors launched the spread of *smart cameras* [26], a compact vision system that integrates into a single unit a camera and a system for digitising and processing the image. These devices typically have the same size as a stand-alone camera and can communicate with the external world in several ways (digital I/O, video output, communication ports, etc.). They are typically employed in simple industrial applications where they can exploit the internal and integrated architecture to provide fast and accurate responses in an efficient and cost-effective way.

1.2.1 2D machine vision

In a 2D vision system, the input is a flat, bi-dimensional image obtained with a digital camera whose core technology is based on one of the following sensors [27]:

- **CCD** (*Charged Coupled Device*): incoming light is converted into electrons through a silicon chip made of an array of photosensitive units; being an analog device, it requires an analog-to-digital converter that transforms the voltage read from each unit to a digital signal used to reconstruct an image;

- **CMOS** (*Complementary Metal-Oxide Semiconductor*): it is an active-pixel sensor, where each pixel unit cell has a photo-detector, and one or more active MOS Field-Effect Transistors (MOSFETs); the complementary and symmetrical MOSFETs architecture amplifies the voltage and reduces the noise, making this sensor immune to high noise and characterized by low static power consumption.

Recently, the CMOS sensor technology outperformed the CCD one [28], whose long-standing main advantage was the capture of images with low noise. With improvements in CMOS technology, this advantage closed as of 2020, and thanks to the lower manufacturing cost, better control of blooming[2], lower image lag, and lower power consumption, CMOS sensors almost completely replaced CCD sensors.

The image obtained with such sensors is rich in information from both a geometrical and semantic point of view. However, it lacks any detail about the shape, the dimension and the distance of the object framed in the picture without resorting to external aid or a priori information. For this reason, it is often employed in those industrial applications that do not require 3-dimensionality, such as bar code reading, object recognition, quality check, presence detection, verification of features or anomaly detection. All of these tasks are usually sensitive to light conditions, which, therefore, must be controlled using suitable artificial lights and illumination.

Nonetheless, using a suitable projection model and a calibrated setup, it is still possible to infer depth information from a 2D image, for instance, in the context of a pick-and-place operation of a known piece over a conveyor belt. In this case, knowing the distance and orientation of a 2D camera with respect to the conveyor surface, the pose of an object of known size can be estimated in the 3D space by geometry calculation. Other strategies involve the usage of a visual marker of known size that is easily recognizable in an image. A marker of this type can be applied to stationary elements or to movable objects or boxes to find their position in space quickly. In both cases, a structured environment is necessary, with interventions and/or constraints on the plant or process. Most recent progress in the field of deep learning introduced a new strategy for solving the problem of single-image depth estimation [29]. Despite the significant inaccuracy in the outcome of such techniques at the industrial level, it is still a useful tool in the world of autonomous navigation and driving, where estimation errors in the order of centimetres (or meters, depending on the application and on the vehicle speed) are not particularly dangerous and, thus, acceptable.

In general, computer vision plays a major role in making the most out of a 2D sensor. Often, precise and meaningful data can be extracted from low-quality sensors if powerful software is employed to process those data. Given their relatively low cost compared to other types of sensors, this is one of the main reasons why 2D cameras are so widespread nowadays in the industry.

1.2.2 3D machine vision

When it is impossible to set up a structured environment, and the tridimensional nature of the scene is crucial to the application, *depth cameras* may be the right solution.

[2] *Blooming* is the phenomenon by which the charge in a pixel exceeds the saturation level and starts to fill adjacent pixels.

These types of sensors, in fact, can acquire a set of distances that can be converted into a 3D representation of an object or scene, typically through a *depth map* or a *point cloud*. The former is nothing but a matrix of pixels, like a 2D camera, where each unit stores a depth value instead of colour information. If also an RGB triplet is associated with this value, the sensor is addressed as *RGB-D camera*. If the intrinsic camera parameters that define the projection model (see App. A) are available, the depth map can be converted into a 3D point cloud where the points are expressed in camera coordinates. A single depth map and the corresponding point cloud are sometimes referred to as 2.5D because they are obtained by a single-view acquisition. The composition of multiple 2.5D views is then necessary to obtain a true 3D data measurement [30].

Currently, depth cameras can be based on four types of technologies (Fig. 1.5) listed in the following, and better described and compared in [31] :

- **Stereoscopy**: it uses two 2D cameras embedded inside a small and compact device, where images are processed to triangulate the position of the measured points of the scene; this technique relies on the fact that the relative position of the two lenses is fixed and known thanks to a calibration procedure; through rectification, corresponding points are efficiently matched and triangulated using the projection model and the intrinsic parameters of both cameras; computing speed is often increased by the employment of Field Programmable Gate Array (FPGA); precision in position estimation is about 5-10% of the distance, depending on the quality of the sensor;

- **Time-of-Flight (ToF)**: it measures the time a wave takes to travel the path from the camera emitter to the object and back to the camera receiver based on maths and physics; different types of waves may be used, but the most common choice is a light signal, produced by either a laser or an LED; precision in position estimation is about 1% of the distance and, thanks to the simplicity of the working principle, they have low computational power requirements; they, however, perform poorly in natural surroundings due to the presence of wave emitted by other sources (e.g. the sun), and with shiny and edgy surfaces due to refraction and reflection phenomena;

- **Structured Light**: it is based on the projection of a known pattern (usually a grid) onto a scene; the deformation of the pattern when striking a surface allows the vision system to compute its depth and shape; it can provide an accuracy up to 1 mm error in depth estimation, but it typically works on short ranges (2-3 meters);

- **Active Infrared Stereoscopy**: it is similar to stereoscopy, but the projection of structured light is used to facilitate the point matching and, therefore, improve the precision of the triangulation and depth estimation;

Compared to two-dimensional image processing, working in three dimensions requires more time and intensive use of advanced processors and software (such as multicore processors, GPUs and 3D algorithms) to manage production line volumes. However, thanks to their ability to reliably capture the extra third dimension, 3D vision systems are more immune to environmental factors that create difficulties for the 2D system, such as brightness, contrast and the need for a structured environment.

Because they work with an accurate digitised three-dimensional model of the target object, the machines that leverage this technology can handle both shape and position.

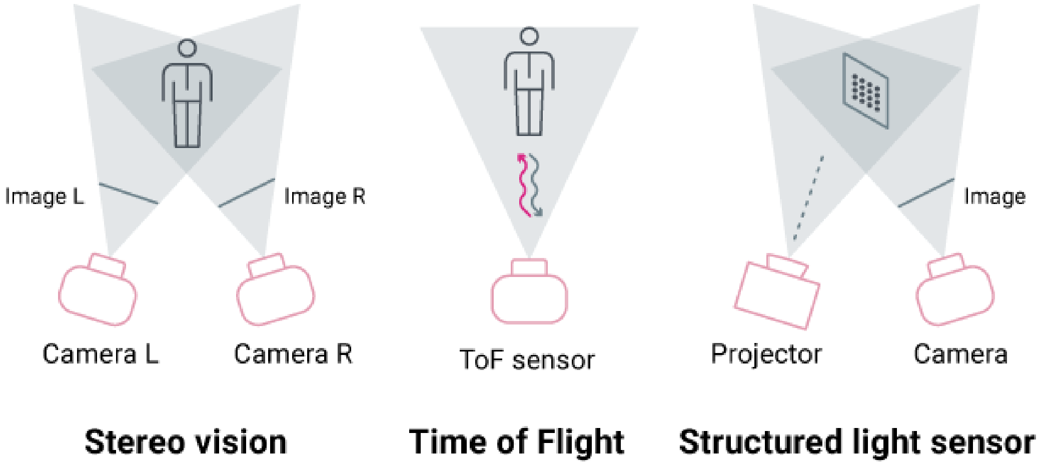


Figure 1.5: Schematic view of the working principle of depth cameras.

Hence, they know the precise pose of an object in space, its exact volume, surface corners, and degrees of flatness, regardless of the conditions of the working environment or whether the object is partially shiny or light-absorbing black. As a result, the 3D vision system can be applied to a wide range of applications where the characteristics of the 2D system are not sufficient, such as:

- thickness, height and volume measurement;
- dimensioning and space management;
- measurement of shape, holes, angles and curves;
- surface detection or assembly defects;
- quality control and verification against 3D CAD models;
- robot orientation and surface tracking (e.g. for welding, glueing, deburring and others);
- container pick-up for moving, packing or assembly;
- object scanning and digitising.

1.2.3 Traditional Computer Vision vs Deep Learning

Generally speaking, traditional Computer Vision (CV) and Deep Learning (DL) applied to images are alternative techniques to solve the same problem: extracting useful information from a picture, be it features, geometries or semantics. However, depending on the type of application, the amount of data being processed, and the processing capabilities, it may be convenient to choose one over the other.

In [32], the authors discuss the main advantages and drawbacks of both strategies, which are reported in short hereafter.

The main conceptual difference between the two approaches is in the workflow, illustrated in Fig. 1.6. While in traditional computer vision, feature extractors are handcrafted and manually selected by an expert engineer, in deep learning, features are directly learned from data (*end-to-end learning*) as underlying patterns observed in the training images. Features are then usually passed to a classifier which generates the desired type of output. By quoting an example from Wired [33]:

"If you want to teach a [deep] neural network to recognize a cat, for instance, you don't tell it to look for whiskers, ears, fur, and eyes. You simply show it thousands and thousands of photos of cats, and eventually it works things out. If it keeps misclassifying foxes as cats, you don't rewrite the code. You just keep coaching it."

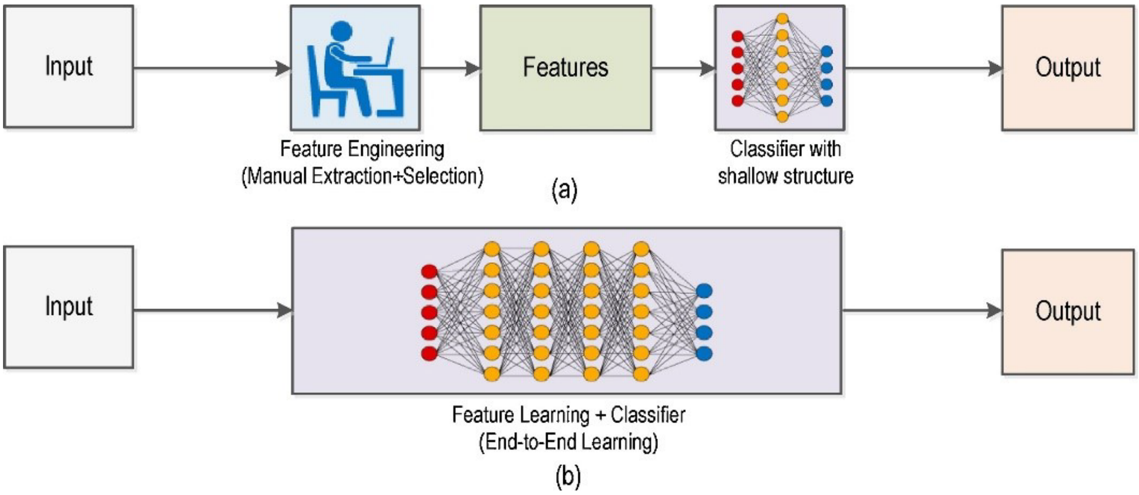


Figure 1.6: (a) Traditional Computer Vision workflow vs. (b) Deep Learning workflow. Figure from [32].

From this concept, it is evident that one of the main issues with deep learning is that it requires a lot of data. Simpler and more specific tasks will need fewer data, but still, a significant amount which is not always available at the industrial level. The idea is that the model cannot generalize without seeing data because it lacks insight into the problem, being it generalized by birth. The risk, then, is to *overfit* the training data, which means shaping a model that predicts well new samples that are similar to the one used for training but may perform very poorly with samples that were never seen before. On the contrary, traditional CV gives full transparency and allows the developer to transfer her/his knowledge about the use case at hand directly into the algorithm. However, choosing the right features to look for in an image may be difficult, especially when the number of classes to classify increases. Moreover, every feature requires manually tuning a set of parameters through a long trial and error process, which often ends up in overfitting anyways. In the context of classification, anomaly detection and object localization under varying conditions, deep learning proved to be the most effective solution, provided that training data cover the whole plethora of scenarios.

Another advantage over traditional CV is that applications using this approach usually require less fine-tuning and expert analysis because Neural Networks in DL are trained rather than programmed. On the other hand, trade-offs with respect to computing requirements and training time must be found.

Nevertheless, DL can sometimes be overkill, requiring long training time, the need for dedicated hardware (e.g. high-powered GPUs), the collection of a sufficiently large training data set, etc. Also, the tuning of the hyper-parameters of the Neural Network can become a cumbersome iterative task which may lead to a significant waste of time. In several well-structured industrial scenarios, traditional CV techniques can solve the same problem more efficiently and in fewer lines of code. Efficiency also plays a major role when the goal is to deploy the software application on an embedded computing unit, given the less demanding power resources that are necessary.

As a naive example, let us consider the problem of the classification of red and green apples on a conveyor belt. DL can be used by collecting enough samples, which, however, would require a possibly long labelling session[3]. The same result can be achieved with a simple colour thresholding technique, which is faster, involves the tuning of a single parameter (e.g. the threshold) and requires no training.

Recently, hybrid approaches have been successfully employed and helped automate some processes while reducing human error. For instance, traditional CV may be the best choice to select a region of interest precisely, and DL to inspect that region. Vice-versa, the result of DL-based object recognition may then be passed back to a traditional CV algorithm to take accurate measurements of the defect size and shape.

In conclusion, there is no clear winner between the two approaches, but the choice is left to the common sense of the engineers. Traditional CV techniques are well-established, transparent and optimized for performance and power efficiency. On the other hand, DL offers higher accuracy and versatility, albeit it requires a large amount of computing resources and data.

1.3 The ROSSINI EU project

Figure 1.7: The ROSSINI logo.

Started on 01/10/2018 and terminated on 01/04/2022, the goal of the *ROSSINI European project*, namely *RObot enhanced SenSing, INtelligence and actuation to Improve job quality in manufacturing*, (see the logo in Fig. 1.7) was to design, develop and demonstrate a modular and scalable platform for the integration of human-centred robotic technologies in industrial production environments. This was achieved by developing innovative technological components and methodologies in all fields related to collaborative robotics (sensing, control, actuation, human aspects, risk assessment methodology) and by integrating all such components in an open platform, ensuring quick ramp-up and easy integration. ROSSINI ultimately aimed at making HRC a viable choice for manufacturers unable to implement it so far due to the regulatory and technological limitations of the technology. A more comprehensive description of the general goals and achievements of the ROSSINI project can be found on the official web page: `https://www.rossini-project.com/`.

To achieve the aforementioned targets, the ROSSINI consortium, led by DataSensing s.r.l.[4], agreed on a set of clear, measurable, realistic and achievable objectives.

[3] The process of associating a *label* to a sample, for instance, "red apple" to an image representing a red apple.

[4] Visit https://www.datasensing.com/ for more details.

One of the main goals was to design the *ROSSINI Smart and Safe Sensing System (RS^4)* with improved detection and tracking capabilities for monitoring the working environment and a safety-graded fusion module for data processing. This groundbreaking sensing technology can radically improve the performance of currently available equipment since closer HRC first needs to rely on highly detailed information based on the working environment. The RS^4 pursued this goal by combining information from several different customised sensing technologies (vision, laser scanner, radar, etc.) to track both the position and the speed of each operator and robot in the scene. Moreover, the miscellaneous information generated by the platform sensing layer was integrated into a single multidimensional image processed in real-time to generate suitable 3D safety regions, or "dynamic safety shells", around each object in the environment. This was achieved by developing a dedicated RS^4 fusion module made of safety field-bus communication and safety sensor modules controller.

Secondly, the *ROSSINI Safety-Aware Control Architecture (RSACA)* for robot cognitive perception and optimal task planning and execution was developed based on RS^4 data. These data, in fact, on top of being piped through safety channels to enhance the safety of the overall work cell, can also be used in a non-safety fashion for high-level analysis, information and feedback. Hence, to enable not only robot sensing but actual perception, suitable data processing techniques based on artificial intelligence were deployed, such as advanced Graphics Processing Unit (GPU) computing and Machine Learning algorithms for image recognition, to obtain a semantic scene map that adapts to dynamic working conditions. This allowed the addition of semantic information to simple geometric maps obtained by fusing sensor data, thus making it possible for the robot not only to visualize images but also to interpret them. Moreover, deploying artificial intelligence techniques for data processing made robot cognition possible. In other words, the RSACA allows robots to optimally schedule the tasks they need to accomplish, paying heed to the trade-off between human operator safety and manufacturing productivity. Each planned action is sent to a dynamic planner that optimizes the trajectory to execute at run-time while considering the Human-Robot (HR) interaction under varying safety conditions in the working area.

Lastly, from a more social and human perspective, a big effort was made to develop a framework for HR mutual understanding in collaborative operations. The ROSSINI platform, in fact, incorporates a human-centred process design that addresses and accounts for human factors like job quality, user experience, trust, the feeling of safety, and liability since the early design stages. Dynamic allocation of tasks allows online changes to the original task planning during the operation. Finally, a profound mutual understanding was realised between robots and people in operation, thereby making HRC more predictable.

The ROSSINI project aimed at using the presented HRC system on three specific uses cases, proving the inherent versatility of such technologies and frameworks:

1. **Schindler use case**: the assembly line of car operating panels is a complex demonstrator due to the high mix/low volume nature of the production, which requires first the kitting of the components, second the assembly itself, then testing and lastly the packaging; these operations are normally performed fully manually; therefore, the goal was to realize a scalable cell in which a product is fully assembled and tested thanks to a close collaboration between a human operator and a robot;

2. **Whirpool use case**: in washing machine production, the assembly of the counterweight is an essential operation; the counterweight is a bulky and heavy piece of concrete used to stabilize the washing unit during spinning operations; the human operator, assisted by a 0-gravity tool, normally picks each counterweight from a container and then positions it onto the washing unit where it must be tightened using screws or springs; in the proposed solution, the list of tasks was cooperatively shared between human workers, robot platforms and related technologies; the solution enables the cooperative operation of the handling, placing and fixing of counterweights, for washing machines; through this cooperation, the human operator may rely on a robotic co-worker that takes over physically demanding tasks and carries out preparations that improve job quality of the human operator and increase the overall efficiency;

3. **IMA use case**: the target was the automatic loading and unloading of raw-material reels to an automatic packaging machine; with weights up to 10 kg, the repeated loading of raw materials can become a source of physical distress for a human operator in the long run; moreover, the operator must always interrupt his/her main job to take care of this short task, leading to a continuous loss of concentration; on the other hand, the number of times that this operation must be executed over a working shift is not sufficient to justify the investment of many standard de-palletizer robots, each one installed next to an automatic machine; a more dynamic solution is thus to be preferred; IMA had already developed and implemented a solution involving a mobile robot system to improve efficiency and ergonomics (see Sec. 2.1); the robot was composed by an Autonomous Guided Vehicle (AGV) equipped with a robotic arm, but showed some limitation, mainly related to safety which were overcame by a similar robotic system now part of the ROSSINI framework.

This Thesis originated from the need to develop a software architecture which integrates all the components and tasks involved in the third use case. Fig. 1.8 shows a high-level abstraction where our main original contributions are highlighted. In particular, most of the work is dedicated to the central "node" of the system, i.e. the ROSSINI mission manager, which is in charge of handling task requests, executing actions, distributing commands to each component of the robotic system and analysing the feedback coming back from them. This node is also where the implementation of the vision-based algorithms presented in this Thesis is deployed. Besides the mission manager, a lot of effort was put into the interfaces with the hardware components of the robot and other elements of the ROSSINI ecosystem. This was done mainly by resorting to pre-existent libraries and packages and by developing the missing blocks that close the communication loop.

While the technical details about the aforementioned integration are discussed in Sec. 2.2 and Ch. 3, it is worth spending a few words about the history of the IMA use case, which is done hereafter in Sec. 1.3.1.

1.3.1 Historical background: from EuRoC to the ROSSINI use case

The *European programme on Robotics Challenges (EuRoC)* was a large-scale integrating project funded by the European Commission within the 7th Framework Programme from 2014 to 2018. Its objective was to create awareness of the core Research and

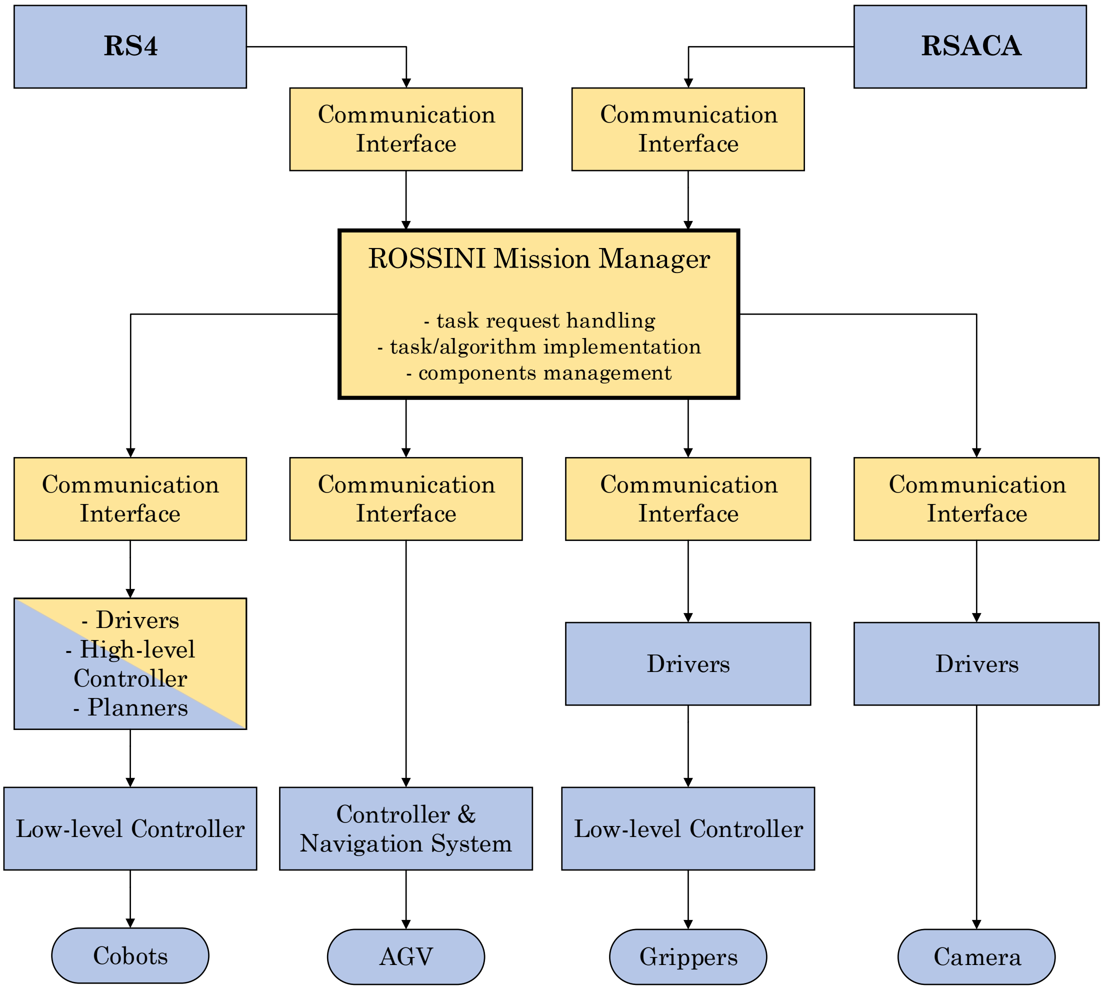


Figure 1.8: High-level abstraction of the ROSSINI software architecture. Yellow blocks are original contributions/implementations, whereas blue blocks are third-party elements which have been integrated. Ellipsoids are hardware components, while rectangles are software components.

Innovation (R&I) issues to be solved in the community through "Grand Challenges". Raising the attention of the stakeholders was a way to bring innovative technologies from research labs to industrial end-users and to exploit synergies across application experts, technology and service providers, and system integrators. To this end, three industry-relevant robotics challenges were launched:

1. re-configurable interactive manufacturing cell;
2. shop floor logistics and manipulation;
3. plant servicing and inspection.

IMA S.p.A.[5], a world leader in the manufacture of automatic machines for packaging of pharmaceuticals, cosmetics, food, tea, and coffee, joined the effort with the Universidade de Aveiro, Portugal (UAVR) to take part in the second challenge with the TIMAIRIS project [34]. The scope of this work was to automatize the blank feeding in tea-packaging machines. Blanks are flat cardboards that, once folded and glued, form

[5] Visit https://ima.it/en/ for more details.

the boxes containing the tea bags. The feeding task is typically accomplished by a human operator by hand and belongs to that category of repetitive, tedious and tiring jobs that may lead to health issues in the long term. Moreover, a human worker may commit unintentional errors, such as picking the wrong pile of blanks.

The solution proposed by the TIMAIRIS team was to use, instead, a robotic system based on a mobile manipulator able to carry out the same job in a flexible, autonomous and collaborative way. Software architecture was specifically designed to take high-level decisions depending on the number of machines to feed and the rate of blank consumption. An onboard eye-on-hand vision system was installed to identify blanks of different shapes and sizes for proper manipulation. A single cobot was used in several compliant modes of operation to robustly and safely manipulate the pile of blanks. Lastly, a safe navigation system provided by the AGV allowed the robotic system to be integrated into an industrial environment populated by humans. In particular, two planar safety areas constantly monitored by laser scanners aboard surround the platform and defined two behaviours:

- whenever a human was detected inside the inner "red" area, which was defined according to the maximum reach of the robot, both the AGV and the manipulator stopped (SMS, see Sec. 1.1.2.1);
- whenever a human was detected inside the outer "yellow" area, which had just a larger radius than the "red" one, only the AGV stopped (SMS, see Sec. 1.1.2.1) whereas the manipulator would continue the ongoing operation if any.

The great results achieved in EuRoC induced the company to continue investing in this technology, which was possible thanks to the *MaXima project*, namely *Multiple Actions for Innovation in Machine Automation* [35], partially co-funded by the Italian Ministry of Economic Development[6]. The project, developed between 2016 and 2019 in collaboration with the Department of Industrial Engineering of the University of Bologna, brought the results and lessons learned in the challenge into a real-case scenario, with the aim of automatizing the change of not only blanks but also raw-material reels, and making the robotic system better suited for an industrial use, and thus, marketable. Given the relevance and similarities with the ROSSINI setup, a more detailed description of the MaXima framework is reported in Sec. 2.1.

In short, starting from industrial solutions available on the market, the same pre-integrated mobile robot as in EuRoC, composed of an AGV and a redundant serial cobot, was selected and equipped with additional sensors and devices tailored to the project goals, such as a custom gripper allowing automatic tool change for the manipulation of different types of objects, namely reels and cardboard blanks. The modular architecture of the communication and decision-making software made the system easy to maintain and extend and provided an easy link to pre-existent automatic machines that can be served by the robot. Suitable computer vision strategies were employed to produce reliable and robust trajectory targets for raw-material manipulation and for interfacing with stationary elements of the robotized production cell. Moreover, feature-matching algorithms were used to assess the validity of the manipulated objects. Despite the promising results achieved during the experimental phases, the moderate success rate (76%) showed in the field tests at the client's factory proved the need for further enhancements to bring the product to its full potential. In particular,

[6] Ministero dello Sviluppo Economico (MISE), Decree 15/10/14, CUP B73D15001070.

increasing the robustness and the task-execution speed were pinpointed as the next goals to achieve the expected marketability and profitability of the proposed solution.

Finally, with these improvement points in mind, the ROSSINI project overtook the use case. A similar mobile platform was redesigned, keeping a few parts and know-how from the MaXima solution, but the overall software architecture, control structure and safety management were completely revisited to achieve better performance and be more resilient. *Resilience* is a recent property of productive systems, which is now one of the main drives of many industrial projects, especially within the European Union. In [36], Zhang and van Luttervelt present an accurate analysis of the subject and introduce the concept of a *Resilient Manufacturing System (RMS)*. They also provide generic and general guidelines that may orient the developers towards the design of new RMSs by birth. Although the MaXima robotic system inherently already tracked guidelines I and II (i.e., redundancy and total function), the ROSSINI framework achieved an even higher level of resilience.

1.4 The SENECA project

The high demand for flexibility in the industry often forces the production chain to change configuration, setup and process parameters. This operation is mostly time-consuming and requires experience and patience to be carried out efficiently. Moreover, due to the peculiarity of each facility where an automatic machine is deployed, it is easy to end up in sub-optimal tuning of the parameters of a particular industrial process. Lastly, the variance in the characteristics of the involved materials represents an additional source of uncertainty that can make a default configuration unsuitable or even ineffective.

Currently, the only way to counteract the undesired behaviour of the machine, poor throughput or a large number of defective output units is through the intervention of a highly skilled human operator. Experience and intuition are generally behind successful corrective actions, which are hard to teach, report and convey to other operators.

The *SENECA* project, namely *Systems Enabling Efficient Cognitive Automation*, investigates the usage of artificial intelligence techniques to automatize some of those processes that currently require high-level skills from a human agent.

The project, co-financed by the Ministry for Economic Development and the region Emilia-Romagna, covers different technology fields such as cognitive automation, advanced sensing, autonomy, simulation and artificial intelligence.

Thanks to SENECA, the results obtained by the ROSSINI project could be internally used as a starting point to solve a completely different task linked to the pharmaceutical industry. The diversity of the context proved the inherent flexibility and versatility of the robotic system developed in the ROSSINI framework, which could be re-used with only minor adjustments and a few task-specific integrations.

1.4.1 Autonomous scanning and cleanliness classification of the surface of a pharmaceutical bin

In the pharmaceutical industry, the containers used for storing, manipulating or mixing powders that will eventually become pills or tablets by compaction are called *bins* (Fig. 1.9).

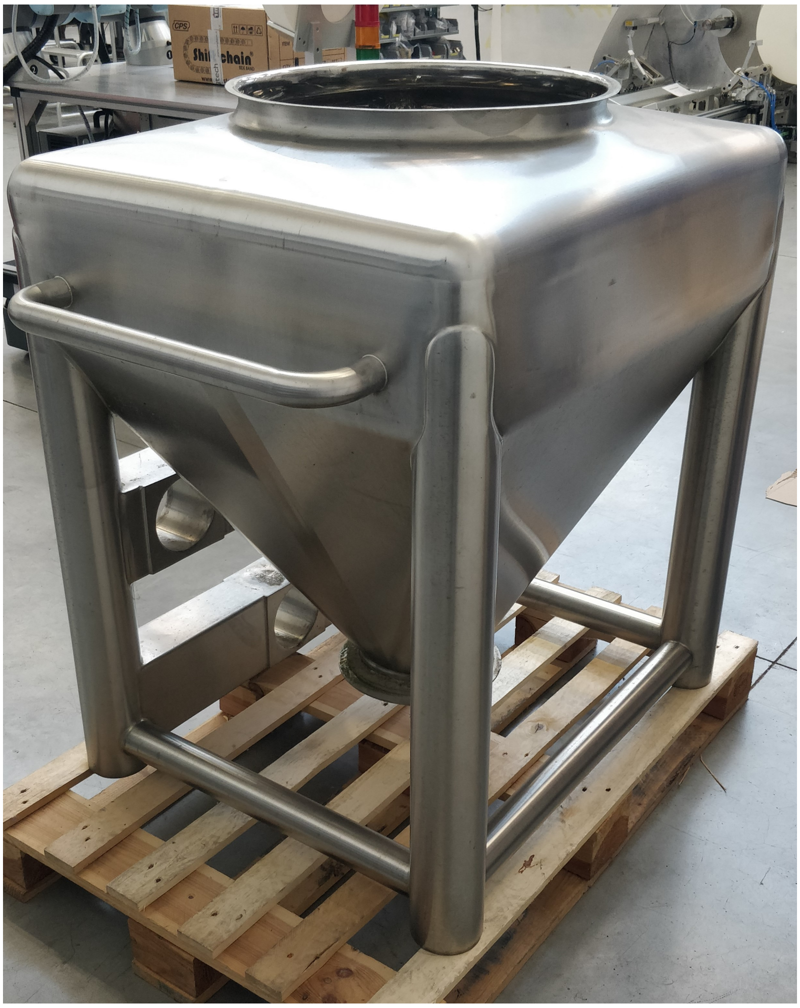


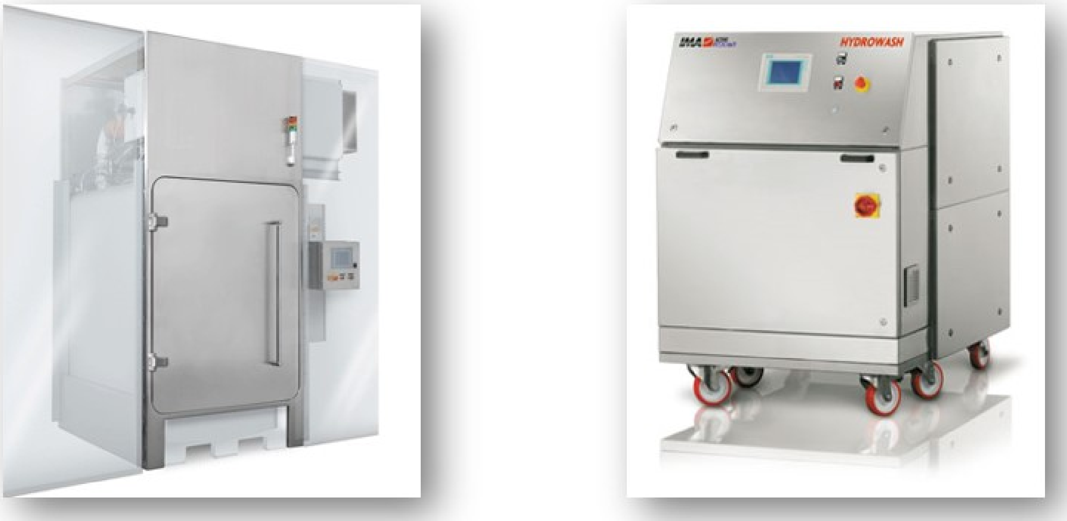
Figure 1.9: A pharmaceutical bin.

Pharmaceutical bins need to be cleaned up to a level known as *critical* because the products contained inside are often incompatible with each other, and their mixture can facilitate the formation of bacterial fauna.

In order to avoid exposure to contaminants harmful to humans, as well as to speed up the process and make it more efficient, it is preferred to automate the cleaning of the bins by means of specialised fully automatic washing booths (e.g. *Ocean Washing Cabin* by IMA Active[7], Fig. 1.10a) or by using a mobile module (e.g. *Hydrowash* by IMA Active, Fig. 1.10b) associated with a cleaning room where an operator manually carries out the cleaning operation with a washing nozzle.

In the master's thesis project [37], the possibility of using a six-axis manipulator to automate the manual operation within the cleaning room (i.e. second option) was demonstrated. A virtual layout of the envisioned setup is shown in Fig. 1.11. This work proved that a solution employing a cleaning robotic arm and a mobile washing module is as economically viable as the equivalent washing booth but with increased flexibility and the possibility to intensify cleaning in the critical areas of the bin. In this context, however, washing trajectories were arbitrarily calculated based on the unique shape of a single bin, which may be a long and tedious job if manually done by the programmer

[7] IMA Active is a division of IMA S.p.A. dedicated to solid dose processing phases, including granulation, tabletting, capsule filling and banding, weight checking, coating, handling and washing. Visit https://ima.it/pharma/brands/ima-active/ for more details.



(a) Ocean Washing Cabin

(b) Hydrowash

Figure 1.10: Bin washing solution by IMA Active.

in charge.

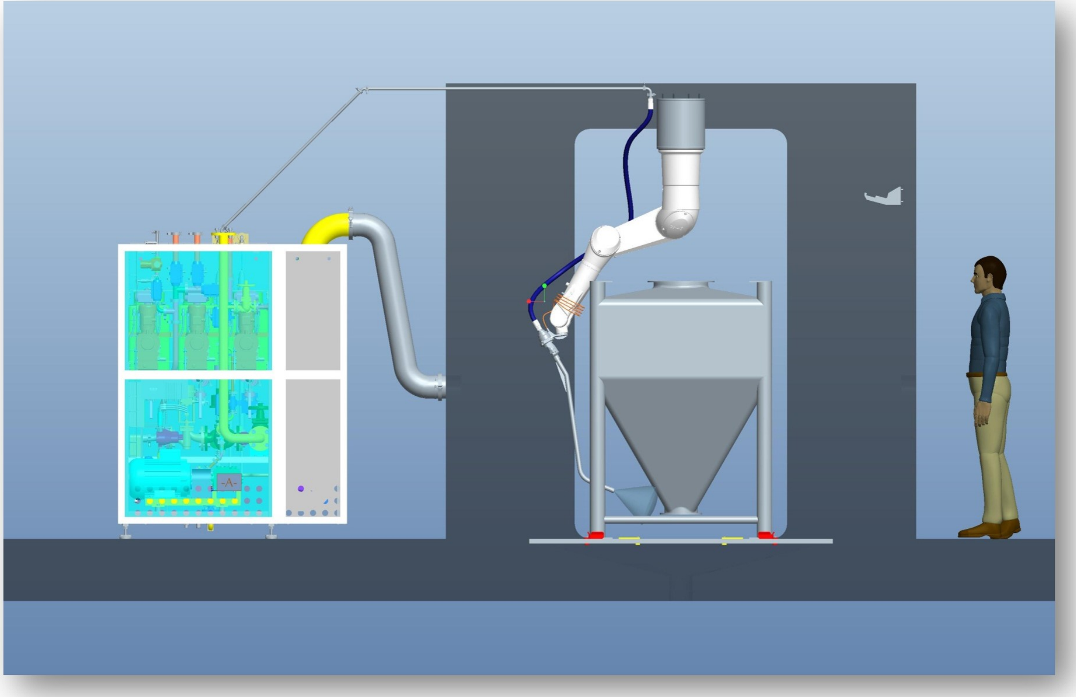


Figure 1.11: Final layout of Hydrowash module + washing cabin with a robotic arm [37].

Based on the preliminary analysis reported in [37], this Thesis aims to extend the flexibility of the envisioned robotic system by introducing vision sensors and advanced algorithms so as to make the operation more intelligent, efficient and versatile.

To this purpose, three phases are outlined:

1. **Bin identification**: assuming a discrete and known number of bins available to

a pharmaceutical company, and assuming that a CAD of the 3D model exists for each one of them, it is possible to autonomously identify which bin has been positioned inside the cleaning room.

2. **Bin surface scanning**: once the CAD corresponding to the bin at hand has been identified, it is possible to calculate the normal directions to the bin surfaces in a selected set of points and exploit the mesh features to generate a suitable inspection/cleaning trajectory automatically; for instance, a path that covers the entire surface with a minimum number of trajectory points and orients either the wash nozzle or the inspection camera in an optimal way according to their relative positions with respect to the bin.

3. **Cleanliness binary classification**: using a pre-trained neural network, it is possible to define with a good approximation whether the area framed in the image is dirty or clean. To do this, it is assumed that this feature is distinguishable by an experienced human eye. This operation also involves collecting sample images showing dirty and clean surfaces to train the algorithm.

In Ch. 4, each phase is addressed in more detail, the chosen strategies are described, and the obtained results are discussed.

1.5 Motivations and contributions

Despite the many examples of mobile robotics applied in industrial scenarios, there still exists a gap between the promising results shown by the academic community and the actual versatility and efficacy of the products that are available on the market for practical use. One of the main concerns of the industry stakeholders is related to safety when introducing a cobot in a fenceless working environment. To guarantee the safety of the humans sharing the space with a cobot, nowadays, market solutions are only used in conjunction with wide safety distances and/or slow robot speeds. The ROSSINI project, especially thanks to the novel concept of RS^4, aims to implement a true HRC, laying the ground for a new working paradigm at the industrial level. The need for a software architecture suitable to the robotic platform employed in the use case chosen to prove this new technology was the main trigger of this Thesis.

Leveraging the results obtained during the ROSSINI use case development, in the SENECA project, we had the chance to further prove the flexibility and reusability of the robotic system. The motivation for this spin-off work arose from a very specific industrial need in the pharmaceutical world that is still not addressed at the market level: the robot-driven automatic cleaning of pharmaceutical bins. The availability of a dexterous mobile robot capable of complex actions and equipped with a vision system, together with the know-how gained throughout ROSSINI, allowed us to achieve tangible and impressive results in just a few months.

Given the industrial context in which this Thesis was developed, its main contributions are to be found in the technical novelties introduced and successfully demonstrated:

- a ROS-based software architecture for the mobile robotic platform developed for the IMA use case of the ROSSINI project;

- the implementation of a robust vision-based procedure for autonomous change of reel with a mobile robot;
- the implementation of a procedure for automatic scanning path generation given a CAD model of a pharmaceutical bin that includes collision avoidance and kinematic constraints;
- the realization of an DL-based binary classifier for surface inspection.

From a more scientific point of view, the main contributions can be found in the following:

- the definition of a robust vision-based procedure for reel-core estimation, which can be easily extended to any problem involving the pose estimation of the centre of a circle in an industrial scenario;
- the adaptation and extension of a procedure for the automatic generation of the scanning path of an object, which can be easily extended to any other inspection/cleaning problem, provided that the CAD model of the object is available.

Chapter 2

Experimental setup

Given the technical focus of the Thesis, a significant space is dedicated to the experimental setup of the two presented use cases: ROSSINI and SENECA. This chapter is organized as follows. In Sec. 2.1, we briefly discuss the hardware and architecture of the MaXima project, which, despite being anterior to this Thesis scope, shares a lot of elements with it, being the actual precursor of the ROSSINI framework, as described in Sec. 1.3.1. The study of the material produced and collected during the MaXima project was crucial in the design stages of the ROSSINI use case and was included in our first publication in the field of collaborative robotics [35]. Next, in Sec. 2.2, the hardware and devices involved in the ROSSINI and SENECA frameworks are described, starting from the robotic platform, which is the common ground for both projects, until each project-specific work cell and tools. In Sec. 2.3, the software architecture is defined, beginning once again with the common features before moving to specific packages and pieces tailored for each use case.

As a necessary disclaimer, it must be said that the electro-mechanical implementation of both frameworks was designed and developed by the hosting company (i.e., IMA), whereas most of the original work of this Thesis lies in the software development, in the algorithms that allow a robust task execution, and in the software architecture choices.

2.1 MaXima hardware setup and architecture

Throughout this Thesis, we use the term *Autonomous Mobile Robot (AMR)* to define the set of main components that characterize a robotic system in the proposed contexts. Moreover, we define *robotized production cell* the set of the elements of a plant, that for the MaXima project are:

1. an automatic packaging machine;

2. a local storage for stocking raw material necessary for a single working shift;

3. an AMR responsible for feeding raw material and monitoring the process.

2.1.1 MaXima autonomous mobile robot

With reference to the numbering in Fig. 2.1, the AMR consists of:

1. an *AGV* that allows the AMR to be moved around the plant, thus capable of autonomously navigating within a given map while avoiding static obstacles and ensuring safety for all human operators sharing the same space;

2. a serial *cobot* for local manipulation of objects;

3. a *gripper*, attached to the manipulator end-effector, that performs grasping and pinching operations;

4. a *vision system* (e.g., 2D camera + laser pointer) able to define targets for the cobot in the proximity of static elements of the plant, as well as to inspect raw material before loading operations.

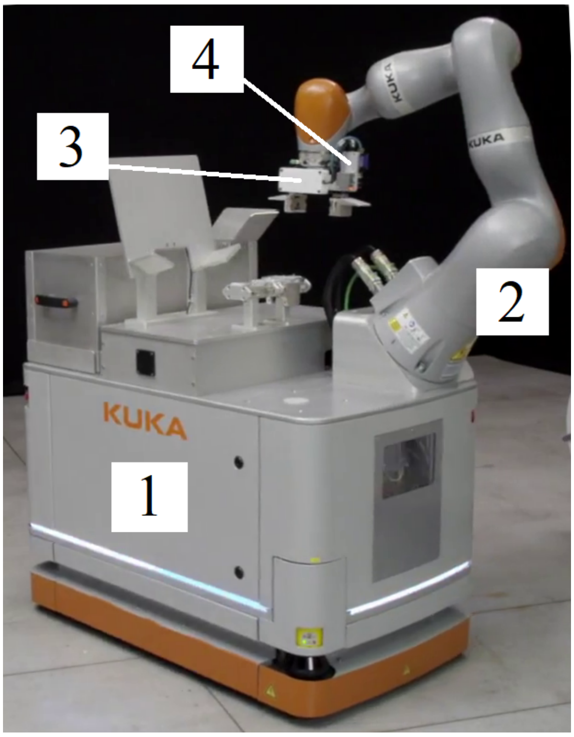
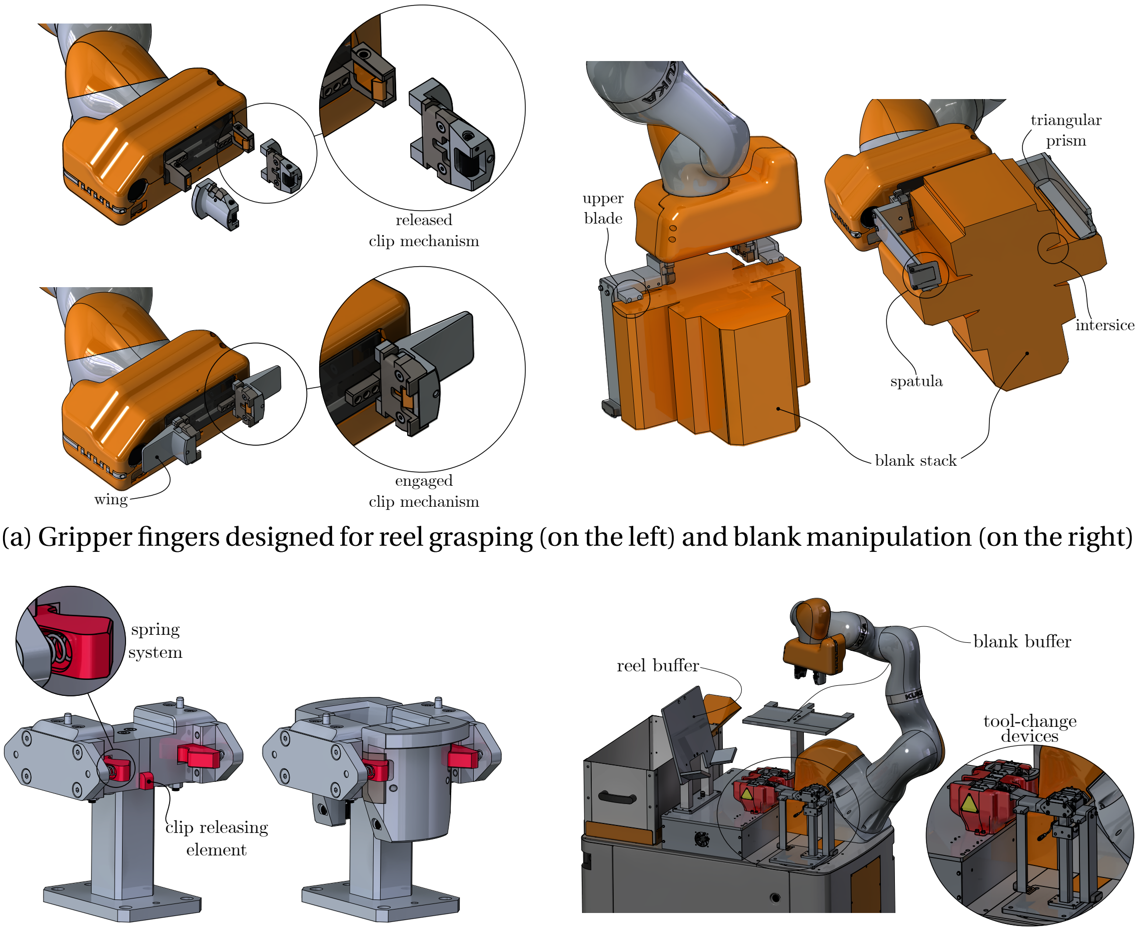


Figure 2.1: MaXima autonomous mobile robot.

The KMR system by *KUKA*, including a mobile platform (from now on referred to as *AGV*) and a lightweight LBR iiwa 14 manipulator (from now on referred to as *cobot*), was selected mainly because of its market readiness, together with the convenience of a fully integrated system, which considerably reduced the set-up efforts on the end-user side.

The LBR iiwa has 7 degrees of freedom, which allows targets to be reached in multiple configurations, thus granting a high level of flexibility. Moreover, each joint is equipped with a torque sensor, which grants a quick and safe arrest in case of unexpected contact. The end-effector is equipped with an industrial electrical gripper by *Zimmer* with custom, inter-changeable fingers, a laser pointer, and an industrial monochrome 2D camera by *Matrix Vision*.

The AGV features Mecanum wheels, which on the one hand, make it a holonomic mobile platform, on the other, constrain it to a flat horizontal surface due to slipping tendency on slopes. Numerous onboard sensors ensure precise and safe navigation



(a) Gripper fingers designed for reel grasping (on the left) and blank manipulation (on the right)

(b) Tool change device (on the left) and example of finger storage (on the right)

(c) Buffers aboard the AGV

Figure 2.2: Custom mechanical components.

within a plant map that can be generated by a preliminary and semi-automatic scanning phase.

The different shapes of the objects to be manipulated by the robotic system required some customized components to be manufactured. As shown in Fig. 2.2a, two different pairs of fingers are employed for reel processing. The first one is able to process both the filter paper and the outer-envelope reels. The round shape of the fingers fits the inner curvature of the reel core in order to ensure a firm grip on the object. The second type of fingers is designed for grabbing tag reels. The only difference is the addition of lateral wings, which prevent the so-called 'streamer effect': an undesired unwinding of the reel during the manipulation due to the sliding of adjacent layers. The triangular prisms fill the blank interstices (Fig. 2.2a) and help to grab the stack firmly. A flexible spatula at each finger's end is employed to ensure that the whole stack is picked up, while the upper blade prevents the stack from excessively bending during the overturning phase. In Fig. 2.2a, the spatula of the right blank finger is hidden from the CAD model for the sake of clarity.

Each finger is designed to allow automatic tool changeover. Tailored mechanical devices mounted on the AGV (one for each finger type - see Fig. 2.2c), thanks to a tooth-shaped spring system, are used to release the clip mechanism that connects the finger to the gripper, detach the fingers and store them until the next changeover (Fig. 2.2b).

In order to handle the objects (i.e. reels/blanks) aboard the AGV, storing devices are needed. In particular, for the reel buffer, a simple plate with two oriented supports allows reels with different diameters to be hosted. On the other hand, the blank buffer mounted on the AGV not only hosts the stack but also helps the flipping operation, thanks to the tailored profile of the plate. Both buffers are shown in Fig. 2.2c.

2.1.2 MaXima work cell

The MaXima work cell consists of a relatively small area (around 100 m^2), including two (semi-)stationary elements: the local storage and the automatic machine.

The automatic machine under consideration is a C24-E[1], a fully automatic machine for packaging tea bags sealed with a double knot, wire and a hard tag. There are neither metal staples nor additional packaging materials to fix the bag to the label and the cotton thread. In fact, label fixing to the thread is guaranteed only by two knots. C24-E, shown in Fig. 2.3, may produce a wide variety of tea bags and boxes, and it can reach a maximum speed of 400 bags/minute with a heat-sealed outer bag and 350 bags/minute with an external envelope closed with a crimped outer bag.

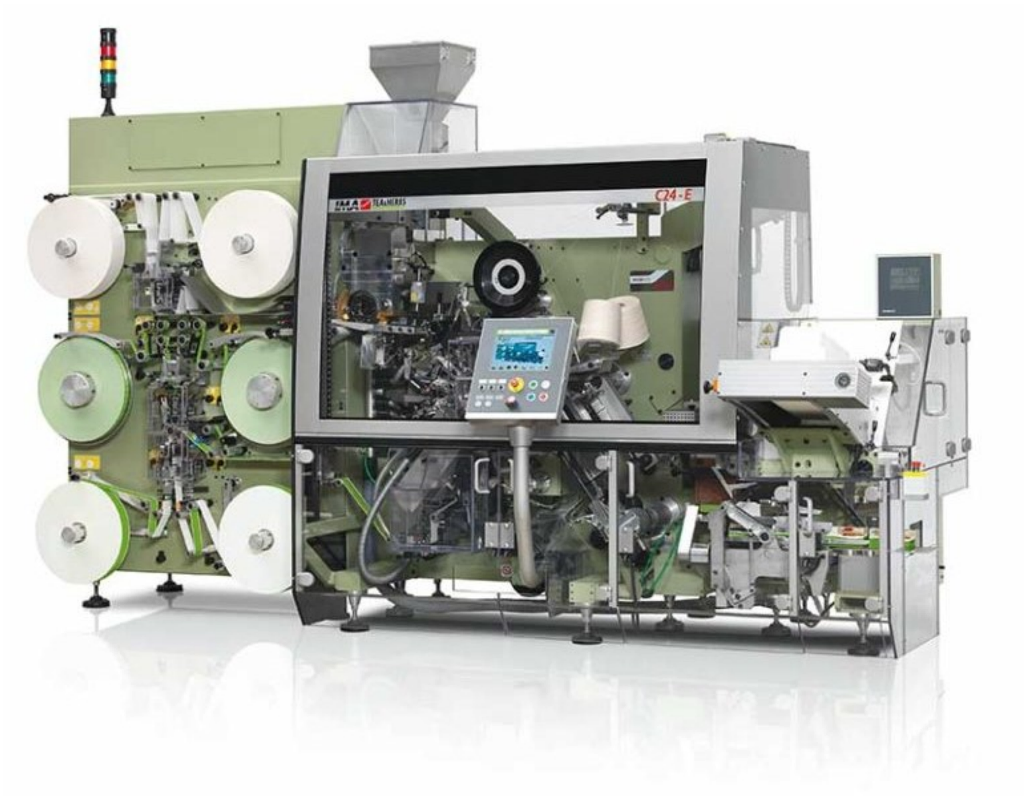


Figure 2.3: C24-E automatic machine.

Fig. 2.4 shows a simplified representation of C24-E and the raw material to be fed to the machine:

- the two top reels (no. 1) are *filter paper*, with a maximum weight of 7.1 kg;
- the two central reels (no. 2) are the *tag paper*, with a maximum weight of 5 kg;

[1] For more details, please visit `https://ima.it/beverage/machine/c24-e/`.

- the two bottom reels (no. 3) are the *outer envelope*, with a maximum weight of 12 kg;
- item no. 4 is a *thread spool*;
- item no. 5 indicates a *stack of blanks*, namely pre-shaped packaging cardboards that will form the boxes containing the tea bags;
- the *hopper* T feeds the tea onto the filter paper, then bag formation begins in wheel R, adding both tag and thread;
- S1 and S2 are *ejection sections* for defective units.

Table 2.1, instead, shows the raw materials consumption rates in the packaging process.

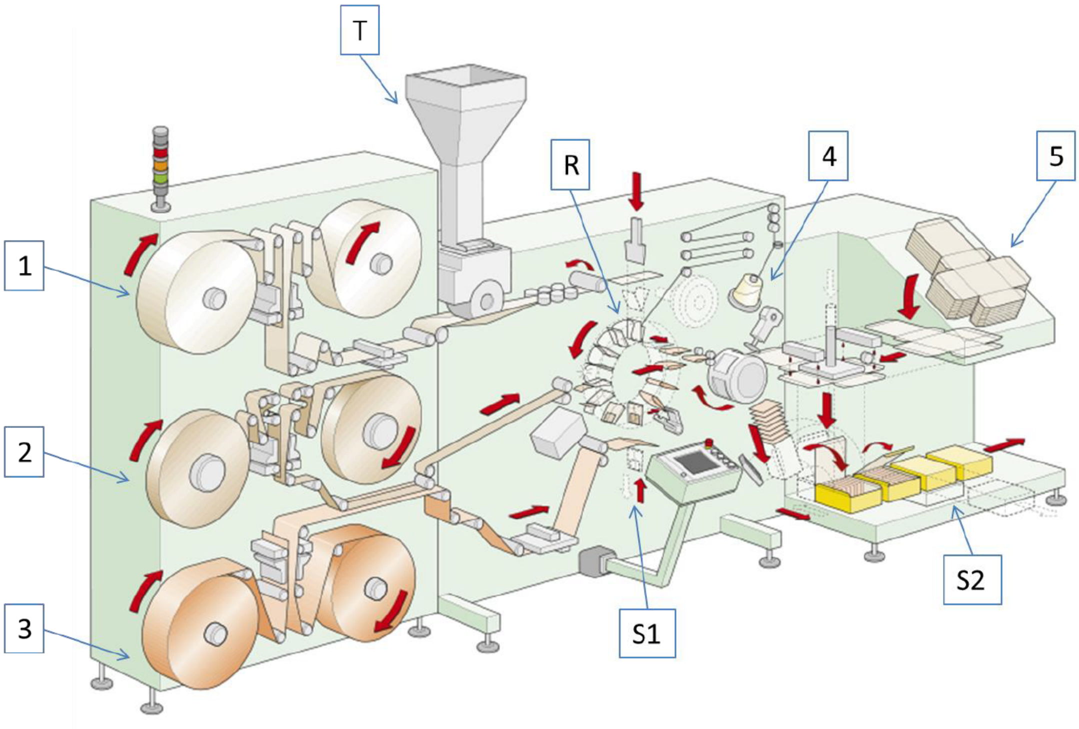


Figure 2.4: C24-E automatic machine schematic view.

Bags/min	Filter paper	Tag paper	Outer envelope		Carton blanks	
			Heat-sealed	Crimped	20 bags/box	25 bags/box
350	100	71	43	56	17.5	14
400	88	63	37	-	20	16

Table 2.1: Average consumption times in minutes for reels and carton blanks.

A *local storage wagon* (or simply *wagon* in the following) (Fig. 2.5) located close to a rack of automatic machines is a common strategy adopted by several IMA clients. It acts as an intermediate storing station between the actual warehouse and the automatic machines and can contain enough reels and blanks for a whole working shift.

In order to decrease the impact of the new robotic system on the plant, this paradigm was maintained. However, the design of the wagon was revisited to simplify the interaction with the AMR. Top shelves are optimized for carrying a pile of filter-paper reels and a pile of outer-envelope reels, slightly tilted to ensure stability. A similar concept is used for the bottom shelves, suitably arranged to allocate a pile of tag-paper reels and another pile of outer-envelope reels. On the side, there is a slot suited for a pallet of blanks, properly stacked to maximise the number of available items. The wagon can be manually moved thanks to lockable castor wheels and brought to the warehouse for a refill. Due to this reason, the wagon position is never perfectly known. Therefore, marker tags are used to help the vision system accurately locate the wagon pose after a first location refinement is carried out by the AGV by exploiting laser-scanner readings and the knowledge of the 3D shape of the wagon.

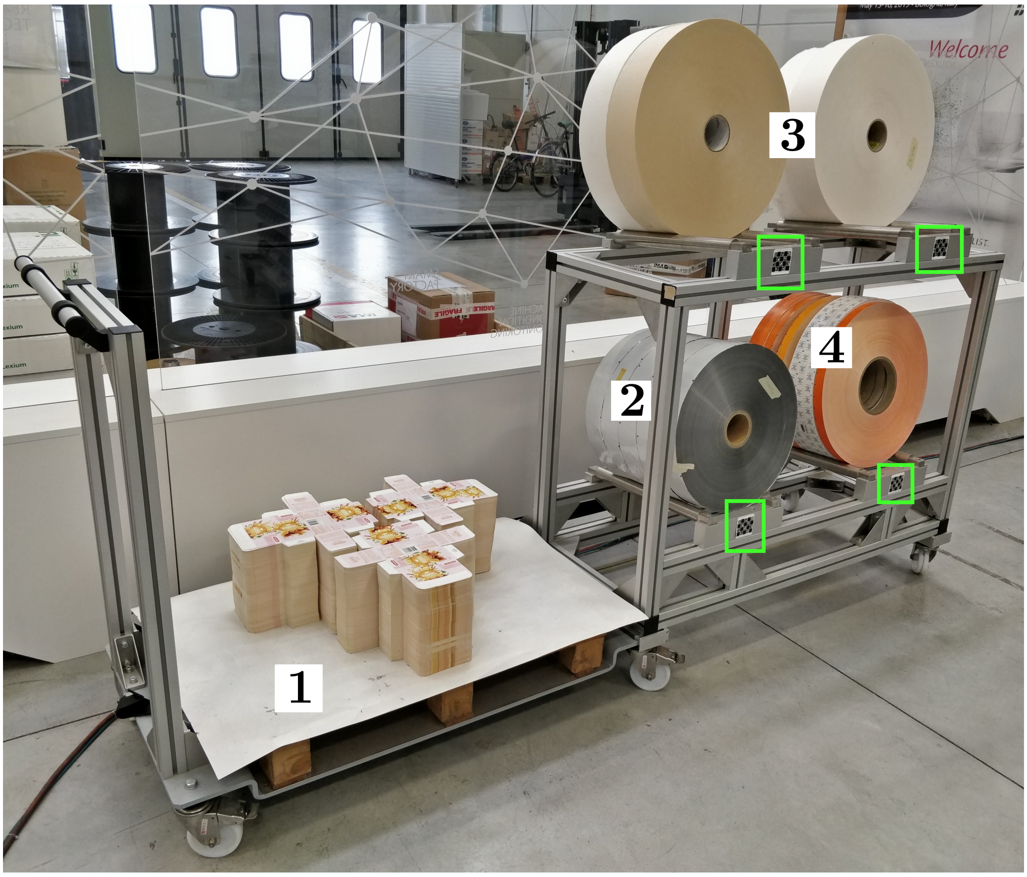


Figure 2.5: The local storage wagon. 1 is the pallet of blanks, 2 are outer-envelope reels, 3 are filter-paper reels, and 4 are tag-paper reels. Green boxes highlight the visual markers of the wagon.

2.1.3 MaXima hardware and software architecture

Fig. 2.6 summarizes the main blocks composing the architecture of the production cell. From a low-level communication point of view, data are transferred within a Local Area Network (LAN), where stationary elements, such as the automatic machine and the navigation server, are connected to a central hub via Ethernet cables, while moving/temporary elements exploit Wi-Fi technology. Each item in Fig. 2.6 is responsible

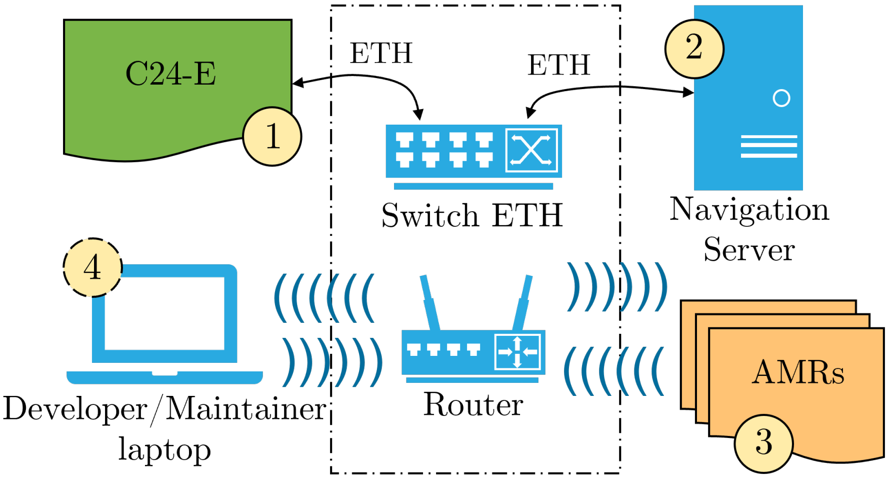


Figure 2.6: Hardware architecture.

for a particular set of tasks[2] illustrated hereafter.

1. **C24-E**: the automatic machine, via Modbus TCP/IP[3] communication, can invoke the help of an AMR and notify of any change of format and/or machine status.

2. **Navigation server**: the server runs on a stationary workstation and sorts all pending requests[4] while monitoring the robot fleet status. Moreover, it is in charge of managing the plant map as well as the definition of all *work locations*, namely charging stations and static positions close to stationary elements where the AMR performs manipulation operations.

3. **AMR**: each element in the AMR is treated independently and covers specific tasks:

   (a) AGV:

   - self-navigation,
   - initial mapping of the plant,
   - safety areas management and monitored stop,
   - enabling the cobot when stationary at a work location;

   (b) cobot (i.e., manipulator):

   - trajectory following, i.e. tracking of way-points manually taught by an expert operator and specifically designed for each task,
   - conversion of information received by the navigation server into elementary operations, such as specific object manipulation, positioning, and switching between force and position control mode,
   - gripper operations management,
   - laser pointer management,

[2] We define as a *task* here a piece of work assigned to an agent, e.g. the AMR.

[3] TCP/IP stands for the combined use of two protocols for transmitting data over the Internet, Transmission Control Protocol and Internet Protocol. The TCP protocol creates the connection between two hosts and manages the delivery of packets from one system to another, while the IP protocol provides the instructions for data transfer.

[4] We define as a *request* here the act of asking for a particular task to be accomplished, formatted in a suitable, computer-friendly way.

- requests to the camera web server,
- safety managements in case of unexpected contact;

(c) camera web server + vision system:

- scanning of markers applied onto stationary elements for accurate relative positioning of the AMR,
- scanning of blanks to infer stack position,
- validity check of raw material.

The different communication protocols native to each device that may request a task necessitate a first layer of decoding/encoding on the navigation server side, which acts as a central control unit for task allocation and request management. With regards to the wireless communication between the navigation server and the AMR, instead, PROFINET was used.

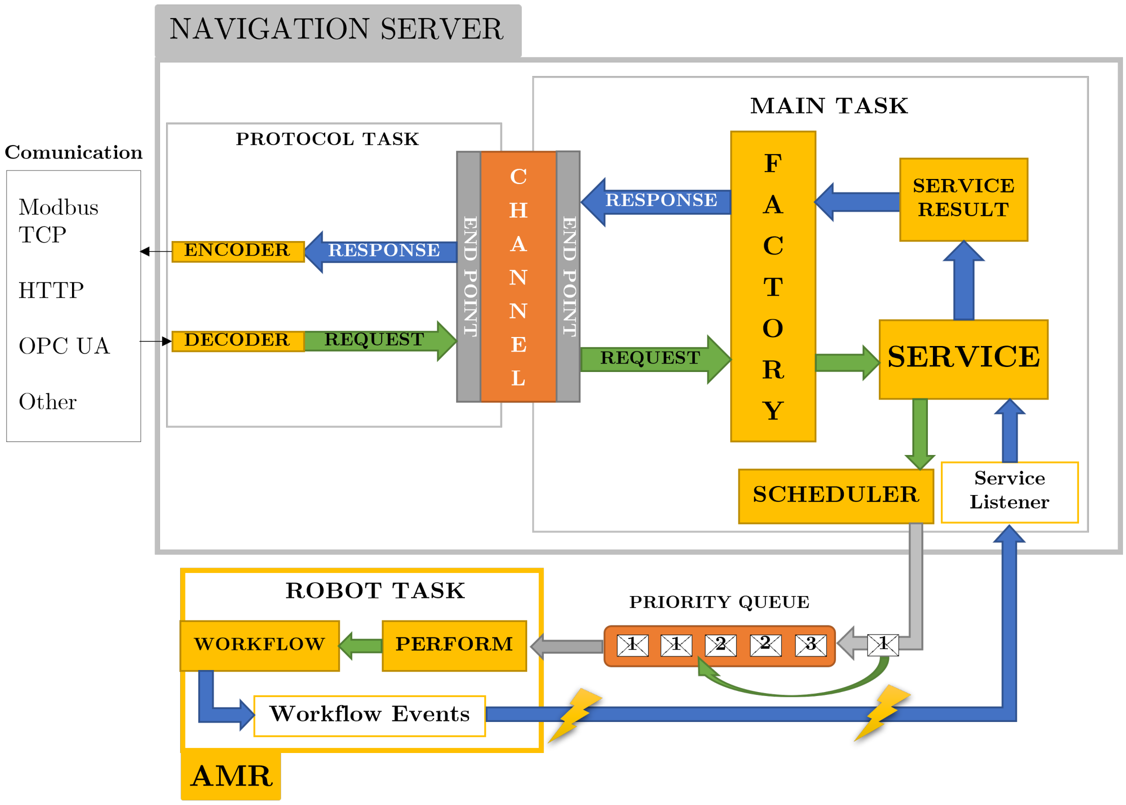


Figure 2.7: Navigation Server Dataflow.

With reference to Figure 2.7, the *Protocol Task* takes care of decoding an external request, for instance, a "Help" request from the automatic machine. The decoded information is passed on to the *Main Task*, and it is used by the *Factory* block to build up a source-independent object, which can be easily interpreted by the system. A *Service* block includes the layout and other important information about the object, such as KMR status and charge, and determines which AMRs can handle the pending request. The *Scheduler*, given the available options at disposal and the current task schedules, appends or inserts the new task in the priority queue of each AMR, which locally parses the information and performs the desired operation while providing feedback throughout its execution. The *Service Listener* continuously reads out the robot's feedback about its status and the status of the active task to update environmental information. Eventually, it forwards the results to the *Factory*, which unpacks the response[5] in a friendly format for the *Protocol Task* to encode it, and to sends it back to the origin of the request.

2.2 ROSSINI & SENECA hardware setup

2.2.1 ROSSINI autonomous mobile robot

The ROSSINI AMR was greatly inspired by the MaXima AMR, leveraging the benefit of having a manipulator installed over an AGV. In particular, the new robotic system is composed of five main elements:

1. two UR10e by *Universal Robots*, a 6DoF serial arm, light-weighted (33.5 kg), with 10 kg payload and reduced dimension (1.3 m reach), and classified as a collaborative robot; the former, addressed with #1, mounts a 3-finger adaptive gripper by *Robotiq* (Fig. 2.8a) to take care of light and sophisticated manipulations, whereas the latter, addressed with #2, is equipped with an industrial 2D camera and a GEH6060IL gripper by *Zimmer* (Fig. 2.8b), and is in charge of inspection/detection tasks and heavy-reel manipulation;

2. a MiR500 by *MiR*, a mobile platform with 500 kg of payload and an advanced navigation system that allows this vehicle not only to orient itself in a mapped environment but also to safely halt whenever an unexpected object appears too close;

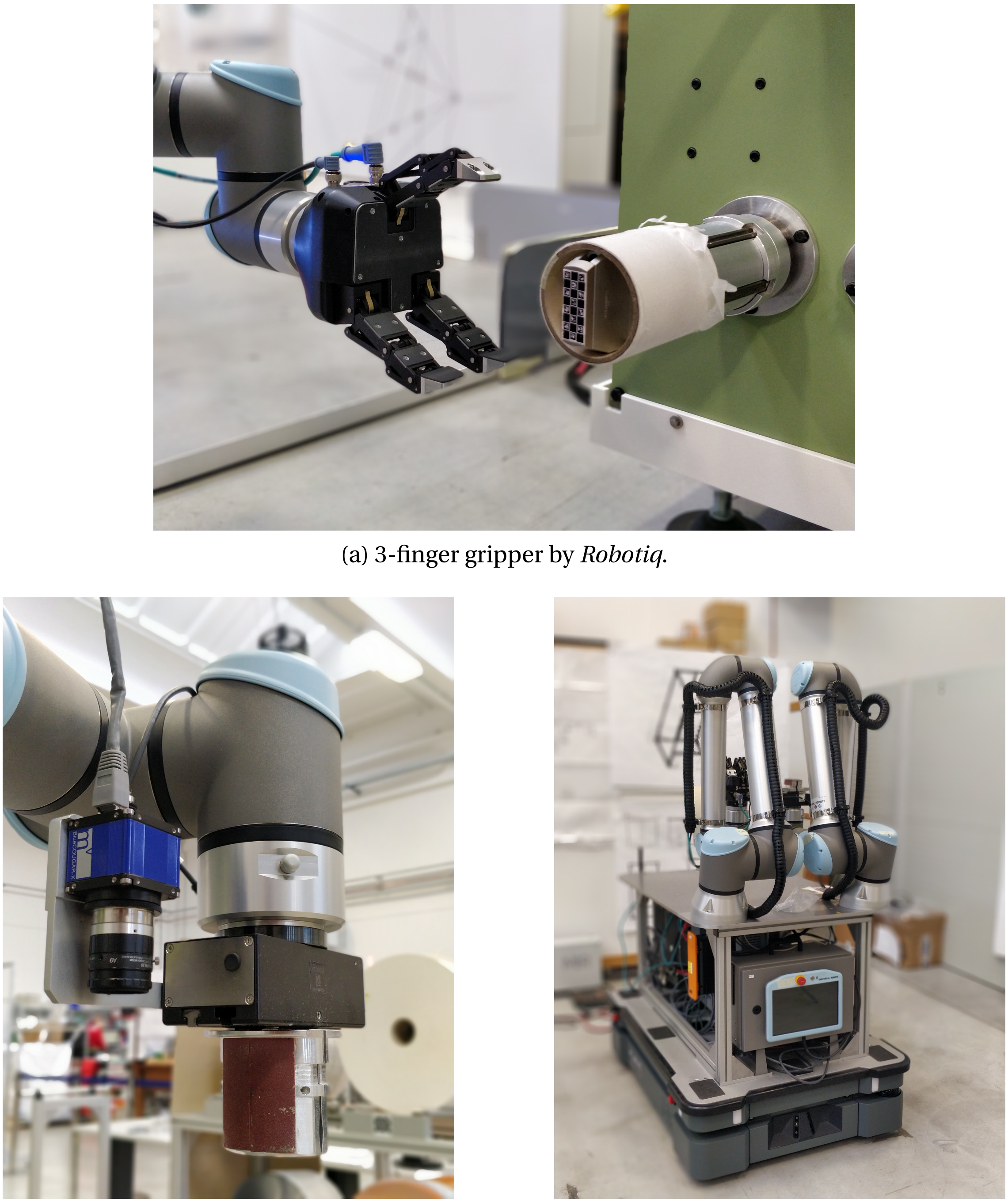
3. an Industrial Personal Computer (IPC) by *B&R*, running Linux Ubuntu 18.04 OS and in charge of all vision-related computations as well as the management of the camera signals and advanced communication with both the URs and the MiR platform;

4. a group of safety PLCs and relays that take care of parsing and distributing emergency signals among all components of the system.

The robotic arms are installed on top of an iron plate fixed on an aluminium frame screwed into the platform's top surface. With the base at 0.9 m height and an angle of 15° with respect to the horizontal, leaning outside the platform footprint, the manipulator can easily reach objects located both at ground level and on high shelves. The space below the iron plate is occupied by the UR controllers and by two racks holding relays, power switches, safety AMRs and the workstation, in addition to all power and data cablings. A photo of the overall assembly of the robotic system is shown in Fig. 2.8c.

Two antennas from *Siemens* are installed at the rear of the platform, one to collect safety signals (and thus safety rated) and a standard one for non-safety data communication. The safety PLCs are then responsible for distributing the safety signal to all the AMR devices according to the situation. In parallel, a standard LAN is used to transfer

[5] We define as a *response* here a reply given as an answer to a task request containing information about the outcome of the performed operation.



(a) 3-finger gripper by *Robotiq*.

(b) IO-link gripper by *Zimmer*
+ industrial 2D camera.

(c) Assembled AMR.

Figure 2.8: The ROSSINI autonomous mobile robot.

high-level commands, feed-backs and general-purpose data among all the workstations involved.

With reference to the MaXima AMR described in Sec. 2.1.1, the choice of a different set of devices has the following motivations:

- the MiR500 navigation and mapping system is considered superior to the KUKA KMR in terms of performance (speed, accuracy, docking), user interface (GUI, web-server) and safety; moreover, from an efficiency point of view, dynamic re-planning is available on the new platform; this means that an unexpected obstacle on the way now triggers the planning of a new path to reach the target position on the map instead of halting the system as it happened with the KMR; lastly, the 500 kg payload leaves more freedom to the integrator and allows to transport heavy objects too;

- the UR10e has a longer arm reach with respect to LBR iiwa 14, even if the lost degree of freedom (6 vs 7) reduces its dexterity; this drawback was resolved by installing two arms next to each other that, together, can cover a larger workspace and possibly co-operate to solve highly dexterous tasks; the 3-finger gripper mounted on the additional arm thanks to its inherent adaptability is then suitable for fine or complex manipulations, such as pinching and grasping of uneven objects; lastly, the UR easily integrates with the MiR communication interface and, most importantly, is cheaper than the LBR iiwa 14 by a significant amount;

- differently from the KMR, which is a built-in system, the fact that the new components are freely integrated leaves a big room for customization, which greatly simplified the development and addition of necessary elements such as the safety PLCs, the IPC and the antenna systems at the expense of more time and human resources to build it.

2.2.2 ROSSINI work cell

Given the common objective, the ROSSINI work cell resembles the MaXima work cell illustrated in Sec. 2.1.2 with a few additions.

First of all, 3D VL-shaped docking markers are placed in a fixed position with respect to each stationary element (e.g., the automatic machine, Fig. 2.9a, and the wagon, Fig. 2.9b) to help the AGV navigation system precisely locate the objects in the map and perform an accurate docking (±5 mm with respect to the $x-$ and the y-axis, ±1° around the vertical axis). The two elements also coincide with the two work locations of the application. An additional point of interest in the map is the charging station, located next to the automatic machine and outside the safety-monitored area.

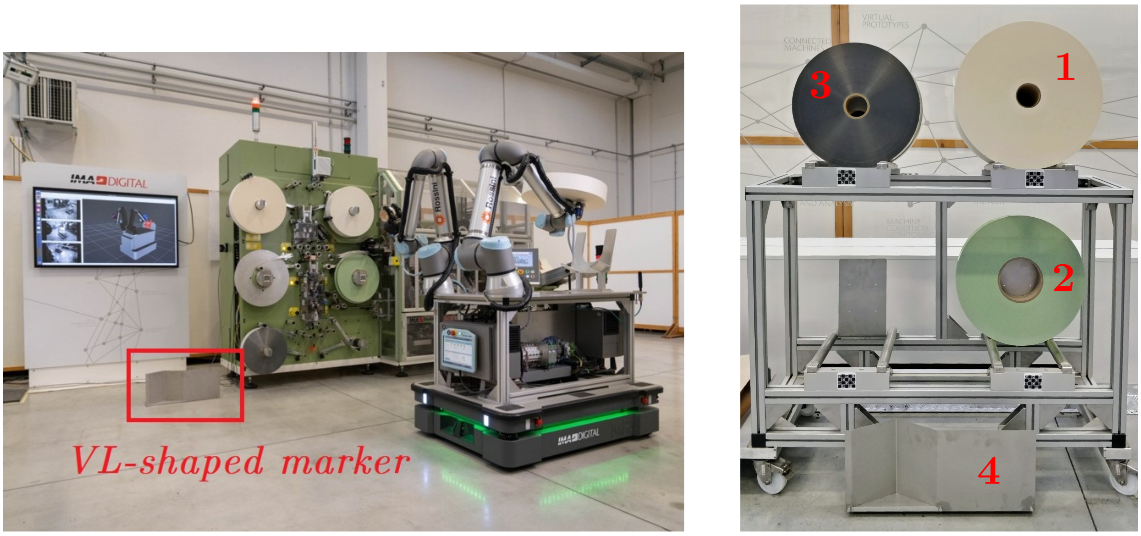
The major change, though, is the integration of the RS^4 safety monitoring system, composed of three safety cameras and a powerful controller for analysing the stream flows and producing both the safety and non-safety signals. The overall setup is shown in Fig. 2.10.

The last addition to the ROSSINI framework is the introduction of a smartphone application, in Fig. 2.11, where instructions, alarms and status feedback can be received and acknowledged by a human operator to enhance HRC.

2.2.2.1 Network

Fig. 2.12 illustrates the ROSSINI network architecture from a hardware point of view. Two main types of channels can be identified: a safety LAN, in red, for elementary safety-rated signals, and a non-safety LAN, in purple, for information, commands and data exchange.

The safety cameras are connected through coaxial cables to a frame grabber in charge of uploading the latest measurements onto a shared memory with the RS^4 controller. Leveraging a GPU, data are elaborated to compute the distance between the



(a) AMR + automatic machine with VL-shaped docking marker.

(b) The local storage wagon. 1 is the filter-paper reel, 2 is the tag-paper reel, 3 is the outer-envelope reel, and 4 is the VL marker.

Figure 2.9: 3D VL-shaped markers in the ROSSINI work cell.

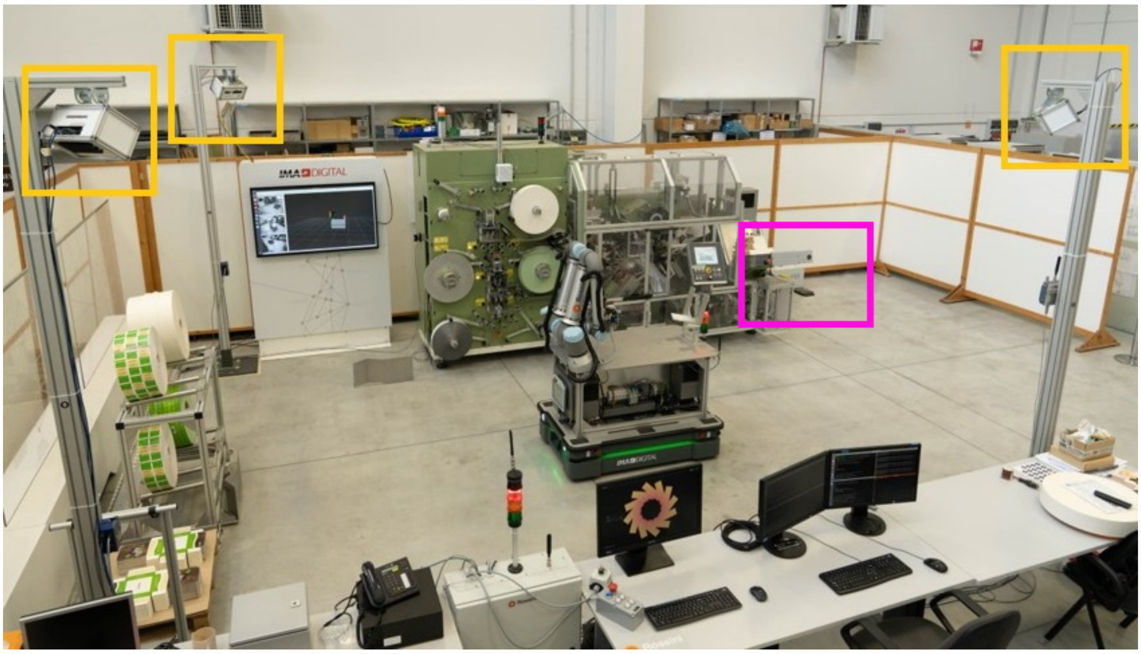


Figure 2.10: Overview of the ROSSINI work cell. The yellow boxes highlight the safety cameras. The magenta box indicates the charging station.

AMR and the closest human operator if any. Depending on this value, a safety signal is emitted through an IO-board connected to a safety PLC that can activate a speed reduction (i.e. SSM, see Sec. 1.1.2.1), a safety-monitored stop (i.e. SMS, see Sec. 1.1.2.1) or confirm that the operation is safe (i.e. no safety mode is necessary). The signal is conveyed to the AMR through a safety Wi-Fi, which internally distributes the signal among its three main components: the AGV and the two manipulators. The RS^4 controller also shares information at a non-safety level by deploying them on a separate

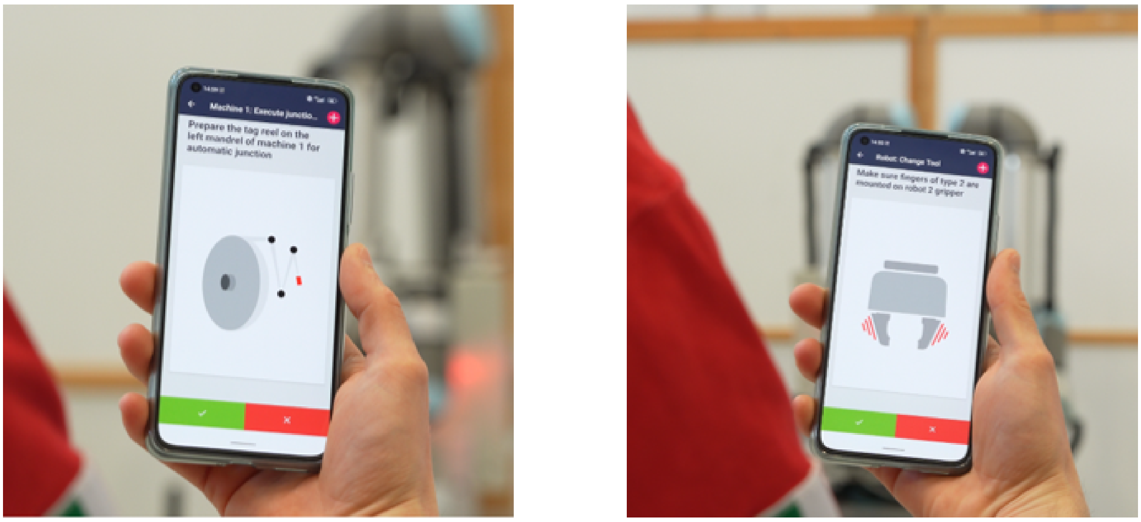

Machine 1: Execute junctio...
Prepare the tag reel on the left mandrel of machine 1 for automatic junction
✓
×
Robot: Change Tool
Make sure fingers of type 2 are mounted on robot 2 gripper
✓
×

Figure 2.11: Examples of instructions on smartphone app.

LAN where the ROS cloud lives, explained in Sec. 2.3. Both the RS^4 controller and the ROSSINI workstation, which is the central unit of the overall framework, are physically connected to a network hub through Ethernet cables. In contrast, the AMR takes part in the LAN thanks to an industrial router that provides standard Wi-Fi to the work cell. Aboard the AMR, the non-safety signals are channelled through a switch that puts in communication the AGV, the controllers of the two cobots, the grippers and the IPC. The IPC, in turn, is connected via an Ethernet cable to the eye-on-hand camera. Lastly, the smartphone is connected through the Internet to the app server that accepts and publishes the instructions in the form of "cards". The commands are generated by software running on the ROSSINI workstation and sent to the app server via the Internet (this requires the workstation to be online).

2.2.3 SENECA work cell

For the SENECA project, the work cell is reduced to a small area that includes only a pharmaceutical bin of medium size and the ROSSINI AMR, without any safety camera monitoring the scene. This is because the technology providers of the safety depth sensors, i.e. DataSensing, had to take back their prototypes as soon as the ROSSINI project ended. For this reason, the safety of the operations carried out in this second project is handled by the AGV when the platform is moving, as it was in ROSSINI, and by the collaborative features of the robotic arm when the manipulator is in motion. In this case, the SMS is not triggered based on the relative position between a human and the cobot, but whenever the safety system of the cobot itself detects an unexpected contact.

In general, in the SENECA use case, we are not interested in the robot safety features because it represents proof of feasibility for a robot which will be constrained in a washing cabin in the future. The robot will be isolated from humans and, therefore, be designed without cooperation/co-existence safety requirements.

Hence, the main reason for this technology choice is essentially motivated by the convenience of using in-house equipment already up and running rather than developing a whole new robotic system from scratch. On top of the consequent acceleration in the experimental phase, re-using the ROSSINI AGV in a completely different context

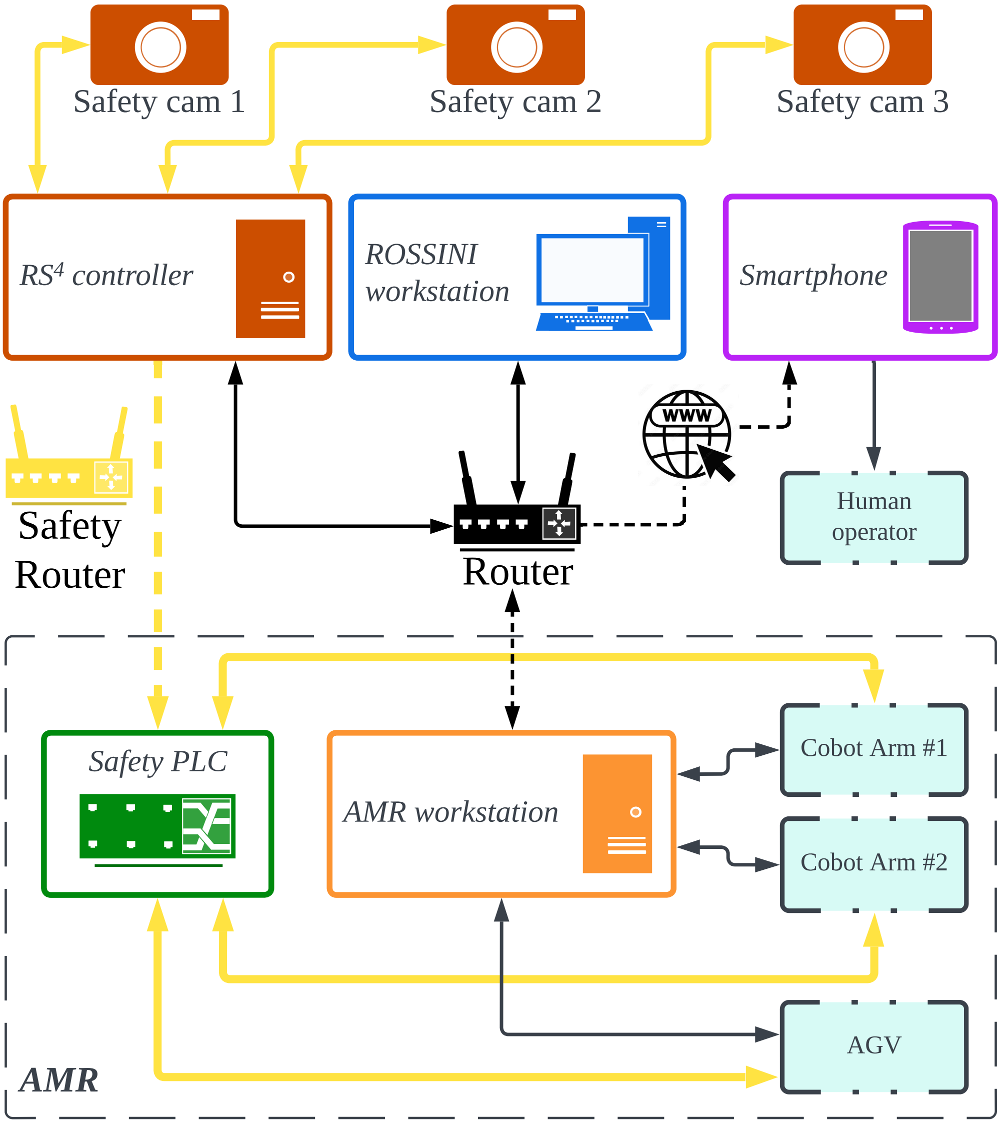


Figure 2.12: ROSSINI network architecture. Dashed lines represent wireless communication. Solid lines represent direct/wired communication. Yellow lines indicate a safety-rated channel for safety signals transportation.

proves its inherent versatility and flexibility, which is aligned with the original scope of the ROSSINI project.

2.2.3.1 Work location

In the SENECA work cell, there is only one work location in front of one of the four sides of the pharmaceutical bin. Differently from the work locations of Sect. 2.2.2, there is no VL-marker for precise docking, meaning that the AGV approaches this position with lower accuracy. This inconvenience is actually desired and better explained in Sect. 4.1.2.

The main dimensions of the bin are shown in Fig. 2.13, which depicts a few poses of the object from its corresponding simplified 3D CAD drawing. More views of the actual bin are illustrated in Fig. 2.14. The bottom and top lid of the bin are removed

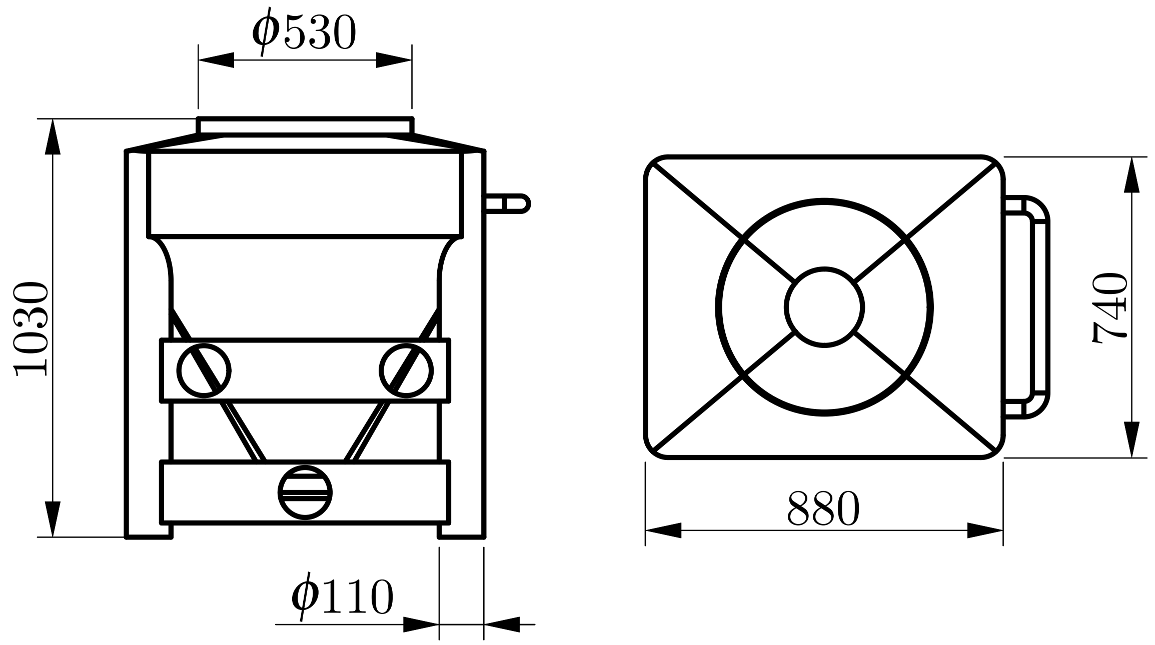
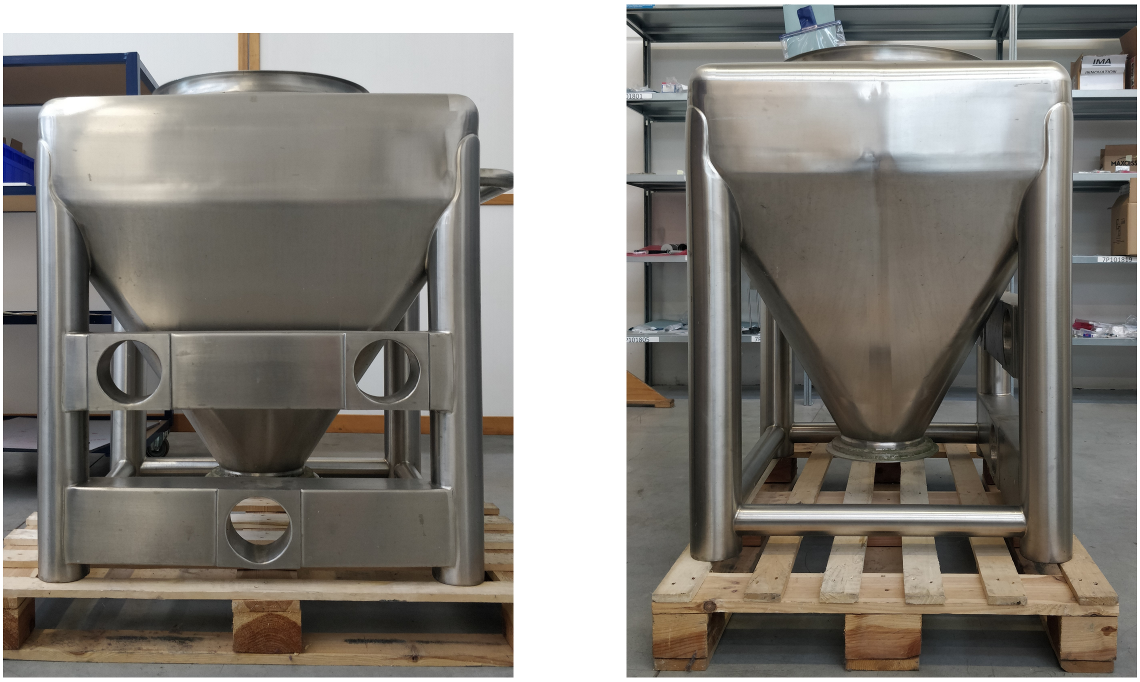


Figure 2.13: Simplified 3D CAD model of the pharmaceutical bin.

as they would be when the bin is being washed to allow the washing nozzles to clean inside.


(a) Frontal view

(b) Lateral view

Figure 2.14: Pharmaceutical bin views.

The bin is made of stainless steel, which is extensively polished on the inside. This means that the internal surface of the bin is much more reflective than the external surface, which appears more opaque. This feature is taken into account to motivate the need for two neural networks to distinguish the dirty areas from the clean ones in the two different scenarios (see Sect. 4.3.1.2).

A ChArUco marker is finally applied to the side of the bin facing the approach direction of the AMR. The marker is used to precisely locate the bin with respect to the base of the arm in charge of the inspection, as explained in Sect. 4.1.2.

2.2.3.2 End-effector tools and sensors

No manipulation is involved in the SENECA use case; thus, both the 3-finger adaptive gripper by *Robotiq* and the custom fingers of the *Zimmer* electrical gripper are removed. In this way, we can reduce power consumption, computational effort and the overall encumbrance of the arms. The *Zimmer* gripper is not removed because it is used as a mechanical support for the 2D camera.

A single additional sensor is required: an RGB camera. Once again, to quickly ramp up the experimental phase, we resorted to the RGB optical sensor of a RealSense D435 by *Intel*, recycled from a different internal pilot project at the hosting company. Another reason to use this device is the availability of ROS-based drivers and packages to connect to the camera stream and to include the 3D model in the planning scene. Fig. 2.15 shows the configuration of the new tool used in this work (fingers are removed because they are unnecessary for this context).

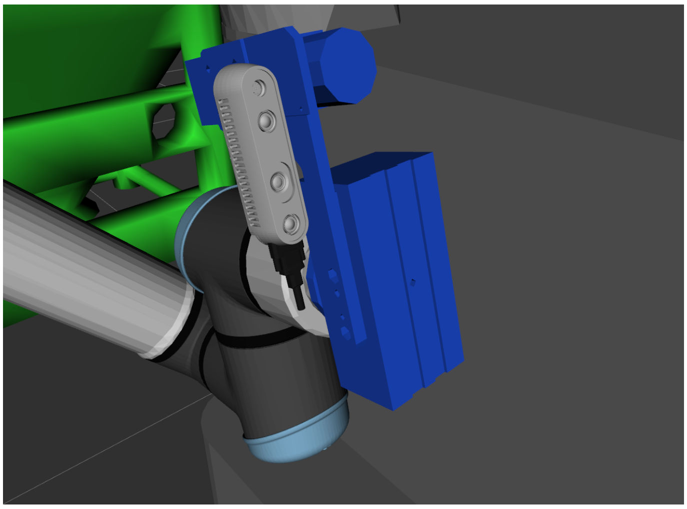


Figure 2.15: Upgraded eye-on-hand vision system.

While the monochrome camera is used to detect and estimate the pose of the ChArUco marker, the RGB camera is necessary to distinguish between dirty and clean surfaces, thanks to the larger amount of information given by the colours and the wider field of view with respect to the industrial camera.

2.3 ROSSINI & SENECA software architecture

In this section, the software architecture related to non-safety communication is discussed.

The architecture is based on *Robotic Operating System (ROS)*, an open-source Software Development Kit (SDK) for robotics applications. It provides libraries and powerful developer tools, from drivers to state-of-the-art algorithms, simulation environments and visualization tools. It is supported by an active global community and has become a standard in the academic field, where it is extensively used for teaching purposes, and it is the basis for many works, from student projects to cutting-edge research. In the past decades, ROS has finally landed in the industry field, where it proved to reduce time-to-market in the development of robotics applications, allowing engineers to dedicate time to the core technology of their business rather than "reinventing the wheel". ROS is multi-domain, from indoor to outdoor, from home to automotive, from underwater to space, and from consumer to industrial. It has helped create billions of dollars in value for start-ups and companies that decided to employ this framework in their business, and the compatibility with ROS is now an added value sponsored by many robotics components vendors. Lastly, ROS is free and grants unfettered access to high-quality software distributed under permissive open-source licenses. The basic working principles and keywords such as *ROS node*, *topic*, *message*, *service* are described in App. B and will be taken for granted in the following.

Regardless of the use case, the *ROS core* is started at boot up on the ROSSINI workstation, laying the ground for communication between the nodes that may be launched thereafter. This also means that the ROSSINI workstation is the first device that must be switched on, being the central hub of the whole network. All visualization and debugging tool are also run from here since it is the only workstation with a graphics driver and monitors. The ROSSINI workstation coincides with the ground station of the work cell. Therefore, all mission requests and task assignments are delivered from here. The workstation is equipped with a powerful GPU (*NVIDIA* RTX 3070) that allows one to train a neural network in a relatively short time or to render a graphic simulation with ease.

The AMR workstation, instead, has much less computational power, and it is mainly used to control the robot and its tools. It is also utilized for the processing of images obtained with the 2D camera for marker detection and reel-core pose estimation, respectively discussed in Sec. 3.2.1 and 3.3.2. In Fig. 2.16, the use-case-independent functional architecture is illustrated.

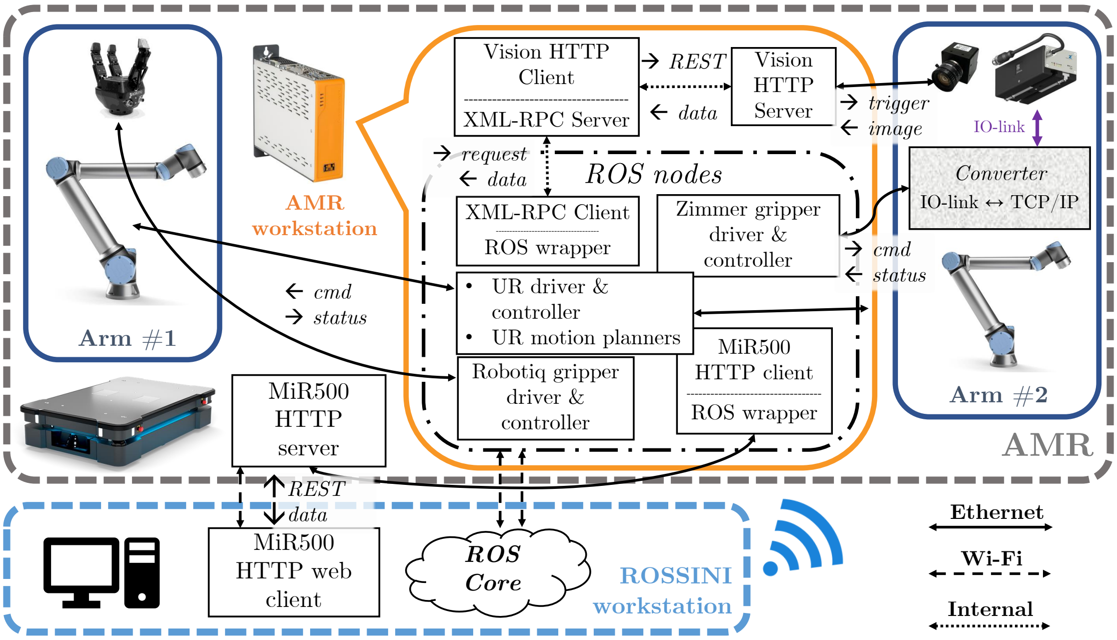


Figure 2.16: Functional use-case-independent architecture.

Each gripper is controlled by a dedicated *ROS driver* that interfaces the hardware, connected through TCP/IP protocol over Ethernet cables, to the *ROS network*. In particular, we have:

- a ROS-compatible driver for the 3-finger gripper by *Robotiq*, downloaded from the open-source repository https://github.com/ros-industrial/robotiq; the node allows to control all degrees of freedom of the gripper through *ROS services*;

- a ROS-compatible driver for the GEH6060IL gripper by *Zimmer*, kindly provided by the manufacturer and soon to be uploaded to a similar open-source repository; the node allows to open/close and set up the parameters and control mode of the gripper through *ROS services*;

The AGV can be controlled from the ground station through a web interface via Wi-Fi for manual operations, as shown in Fig. 2.17. During a robot mission, instead, commands and feedback are directly exchanged between the AGV and the *ROS cloud* leveraging the web HTTP[6] server running on the AGV controller, accessible through REST APIs[7].

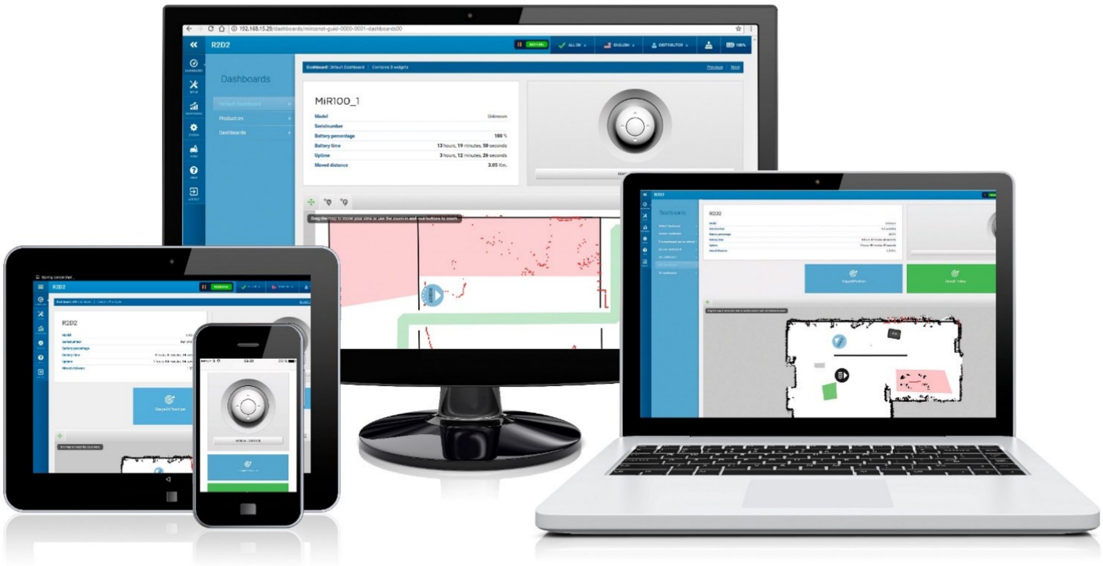


Figure 2.17: MiR web interface. Source: MiR robot reference guide SW 2.6.0 rev. 1.9.

The same HTTP web server that was developed during the MaXima project and controls the 2D eye-on-hand camera while providing basic image-processing functionalities is re-used here. The server includes the driver to physically connect and communicates with the camera through TCP/IP protocol over an Ethernet cable. Moreover, it is possible to grab a new frame or estimate the pose of a ChArUco board with respect to the camera frame through HTTP REST APIs, provided that the camera is calibrated (see Sec. 3.2.1). A module that bridges the communication between the vision web server running on the AMR workstation and the UR10e #2 operative system, namely the *XML-RPC*[8] *server/HTTP client module*, was developed during preliminary

[6] HyperText Transfer Protocol (HTTP) is an application-level protocol used as the main system for transmitting information on the web, i.e. in typical client-server architecture.

[7] Web Application Programming Interfaces (APIs) or services conforming to the REpresentational State Transfer (REST) architectural style.

[8] XML-RPC is a protocol used in IT that allows for RPCs (RPC) via the Internet (HTTP) using XML as encoding.

operations at the beginning of the ROSSINI project, thus before ROS came into play. The reason behind this choice, as opposed to building an acquisition pipeline from scratch, stemmed from the decision to use built-in communication interfaces only on the UR side. Each UR, in fact, makes available a client XML-RPC on startup. By plugging an Ethernet cable into the control box and instantiating the client, it is possible to call external functions that are available on the server side, running onto the AMR workstation in our case. The XML-RPC client is instantiated by defining the IP address and the port at the beginning of a Polyscope[9] program. In this way, we could outsource heavy calculations or complex tasks, such as all computer-vision-related duties. This passive module, therefore, remains idle while continuously listening for requests. Whenever it needs images or camera-related information available as functions on the HTTP web server, it generates the corresponding REST request. By doing so, the reusability of the previous HTTP web server is enhanced while keeping it untouched and leaving room for the implementation of new tasks not directly related to the original purpose of the HTTP vision web server. The following vision-related functionalities are available on the XML-RPC server side:

- calibration utilities (pattern detection, sample collection, calibration routine);
- ChArUco-marker-pose estimation and relative positioning with respect to the estimated marker location;
- reel-pose estimation (raw-frame grabbing, reel detection and localization).

A custom wrapper was developed through an XML-RPC client inserted in a *ROS node* to interface the XML-RPC server module with the *ROS cloud*. The *ROS node* guarantees the reusability of the vision-related software developed so far, both on the HTTP web server side and the XML-RPC server/HTTP client side, by instantiating a simple *ROS service*.

The cobots are controlled through the official ROS-Industrial[10] *Universal Robots* drivers and controllers. Based on this resource, a new *ROS package* containing geometric, visual and control information specific to the ROSSINI AMR was created. This allows the user to control one arm or the other depending on the executed task while being aware of the overall status and encumbrance of the system. This information is particularly useful for planning collision-free trajectories.

To this purpose, ROS *MoveIt* library [38] is used, whose high-level system architecture is illustrated in Fig. 2.18. The library incorporates the latest advances in manipulation, motion planning, 3D perception, kinematics, navigation and control and has become the common ground for many academic and industrial projects involving a robotic kinematic chain, usually a serial manipulator. From this valuable resource, the functionalities related to kinematics, motion planning and collision checking are especially used. The concept of *planning scene* is first introduced as a representation of the world around the robot and the state of the robot itself. A virtual digital twin of the AMR is, in fact, generated using the meshes obtained from the CAD models of each component of the system. This also allowed us to test routines without connecting to

[9] UR proprietary programming environment, a Graphical User Interface that allows the user to manually control the robot and/or write a program directly on the touchpad attached to the control box.

[10] ROS-Industrial is an open-source project that extends the advanced capabilities of ROS software to industrial-relevant hardware and applications.

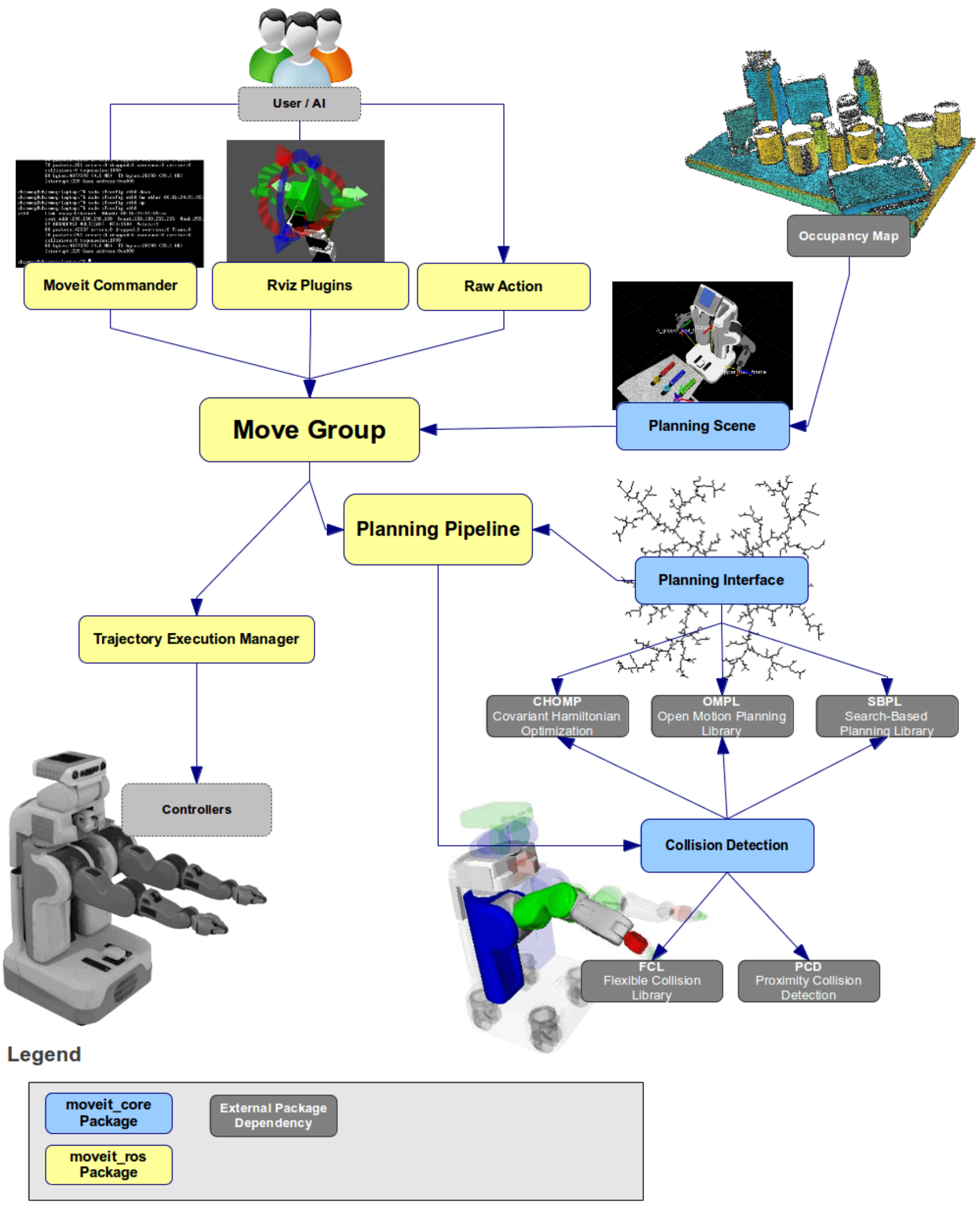


Figure 2.18: MoveIt high-level diagram. Source: `https://moveit.ros.org/documentation/concepts/`.

the actual hardware at the early stages of software development. In the planning scene, as schematized in Fig. 2.19, the *world geometry monitor* keeps track of the world scene, which can be updated with new static collision objects or objects attached to a moving part of the robot (such as the gripper). In parallel, the *scene monitor* keeps track of the robot state in terms of joint position and coordinate systems. When the hardware is connected, this information comes directly from the UR robot drivers; otherwise, it is simulated.

Given a target in either joint or Cartesian space and an updated planning scene that incorporates the starting robot state, a collision-free trajectory is dynamically generated on the fly based on the *Open Motion Planning Library (OMPL)* [39]. Being a pri-

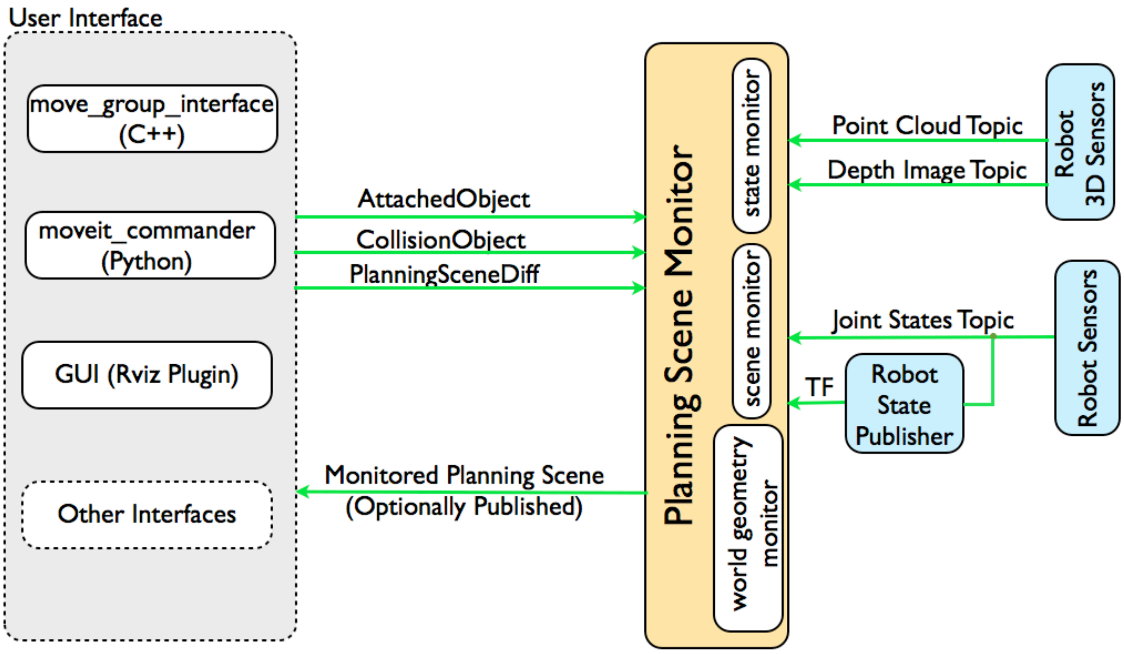


Figure 2.19: MoveIt planning scene pipeline. Source: `https://moveit.ros.org/documentation/concepts/`.

marily set of abstract stochastic sampling-based motion planners, MoveIt provides the back-end to work with robotics-related problems. The motion-plan request specifies what the motion planner should try to attain in terms of the final goal, position and/or orientation constraints and joints constraints. Collisions, including self-collision and attached objects, are checked for by default, leveraging the *Flexible Collision Library (FCL)* [40]. In general, OMPL planners favour speed in finding a solution path over quality and efficiency. Hence, there is no guarantee that a global optimum is found nor that the same solution is obtained from the same start and goal configuration since the algorithms in OMPL are probabilistic. Some of them, however, can give theoretical optimality guarantees, but usually only asymptotically. It is the case for the planner selected for our system: RRT-Connect [41], whose basic idea is to grow two RRTs (Rapidly-exploring Random Trees), one from the start and one from the goal, and attempt to connect them. For the sake of simplicity, this package will be referred to as *dynamic planner* in the following.

While large motions are generated and managed by the dynamic planner, short rectilinear motions that involve an external contact, such as the insertion of a peg in a hole, cannot be planned in this way. Intuitively, this is because planned trajectories must be collision-free, and a safety margin between the meshes of the components spawned in the planning scene must be considered to account for modelling and positioning errors. A different node based on the *moveit_servo ROS package* was developed to account for this limitation. This node, addressed as *moveit_servo_wrapper*, directly connects to the state of the controller's arm and allows to translate the Tool-Centre Point (TCP) of the cobot (or any reference frame attached to it) along the direction of a specified axis. The reference axis is expressed in the cobot base coordinate system, which is static during cobot motions. The rectilinear motion terminates when the desired distance is covered or when a contact is detected, thanks to the force sensor embedded in the cobot terminal link. The desired direction and stop conditions are used to emulate a joystick command that controls the translational velocity of the TCP. This command, in turn, is fed to the *moveit_servo* node, which converts the translational velocity into joint speed set-points for the cobot. This node also handles singularities, decreasing the original target translational velocity when approaching one and stopping the robot when too close to it. Additionally, the wrapper implements a simple second-order ramp to modulate the speed at the beginning of the motion and at the end of it (only when it stops because of the target distance reached). This is done to avoid that a large inertia due to a heavy payload attached to the end-effector is mistaken for a collision and triggers a safety-monitored stop.

To interface with all aforementioned packages, a custom package called *rossini_-mission_manager* was also developed within the ROSSINI project and was then reused in the SENECA project. It includes a few Python libraries to easily interact with the AGV, the grippers, the XML-RPC wrapper (and, thus, the 2D camera and related features), the planning scene and the motion planners, both the rectilinear one and the dynamic planner. Based on these utilities, *missions* can be built as a combination of actions involving one component of the AMR at a time. An abstract mission is provided as a set of sub-tasks, where each task, if successful, leads to the next one. On the contrary, if it fails, it triggers a suitable and customizable fall-back action.

In Fig. 2.20, a functional scheme of the basic software architecture for the ROSSINI AMR is displayed.

2.3.1 ROSSINI architecture and resources

For the ROSSINI project, the software architecture outlined in Sec. 2.3 was extended to interface with the RS^4 safety-camera system. On top of the safety communication, which is ROS-independent, details about the safety state, the distance between human and robot, the image streams and the point clouds of the moving elements in the work cell are broadcast to the *ROS cloud*.

These pieces of information can be used for debugging and monitoring purposes, but also to determine the behaviour of the cobots during motion. Despite the fact that speed reduction and safety-monitored stops are managed by the cobot safety system disregarding the high-level controller (e.g. ROS), safety information on non-safety channels is utilized to prevent undesired interruptions of the process when the cobot is halted or slowed down by the RS^4 system. For instance, if an SMS is triggered during a rectilinear motion, this information is used to turn off the joystick-like commands and apply the second-order ramp filter again when the execution is resumed. Similarly, the same information is used to pause the motion of the grippers because they do not have a safety port or communication channel. This mimics the behaviour of an SMS, but it is not officially safety-rated. In general, to comply with collaborative operations, each device must be collaborative by birth; that is, it must have dedicated safety channels and low-level safety features that can handle an SMS or an SSM signal. In the ROSSINI setup, only the AGV and the two cobots fall under this category.

The RS^4 needs to know when the AMR stops in front of the stationary elements of the work cell (e.g. wagon or C24-E) because only under that circumstance it is active and takes over the safety monitoring. To this end, when the UR drivers are launched, also a node that broadcast the AGV odometry[11] to the *ROS cloud* is started. The node synchronously publishes the AGV state and AGV mission status and handles the mis-

[11] The use and fusion of data from exteroceptive and proprioceptive motion sensors to estimate the change in position over time relative to a starting location.

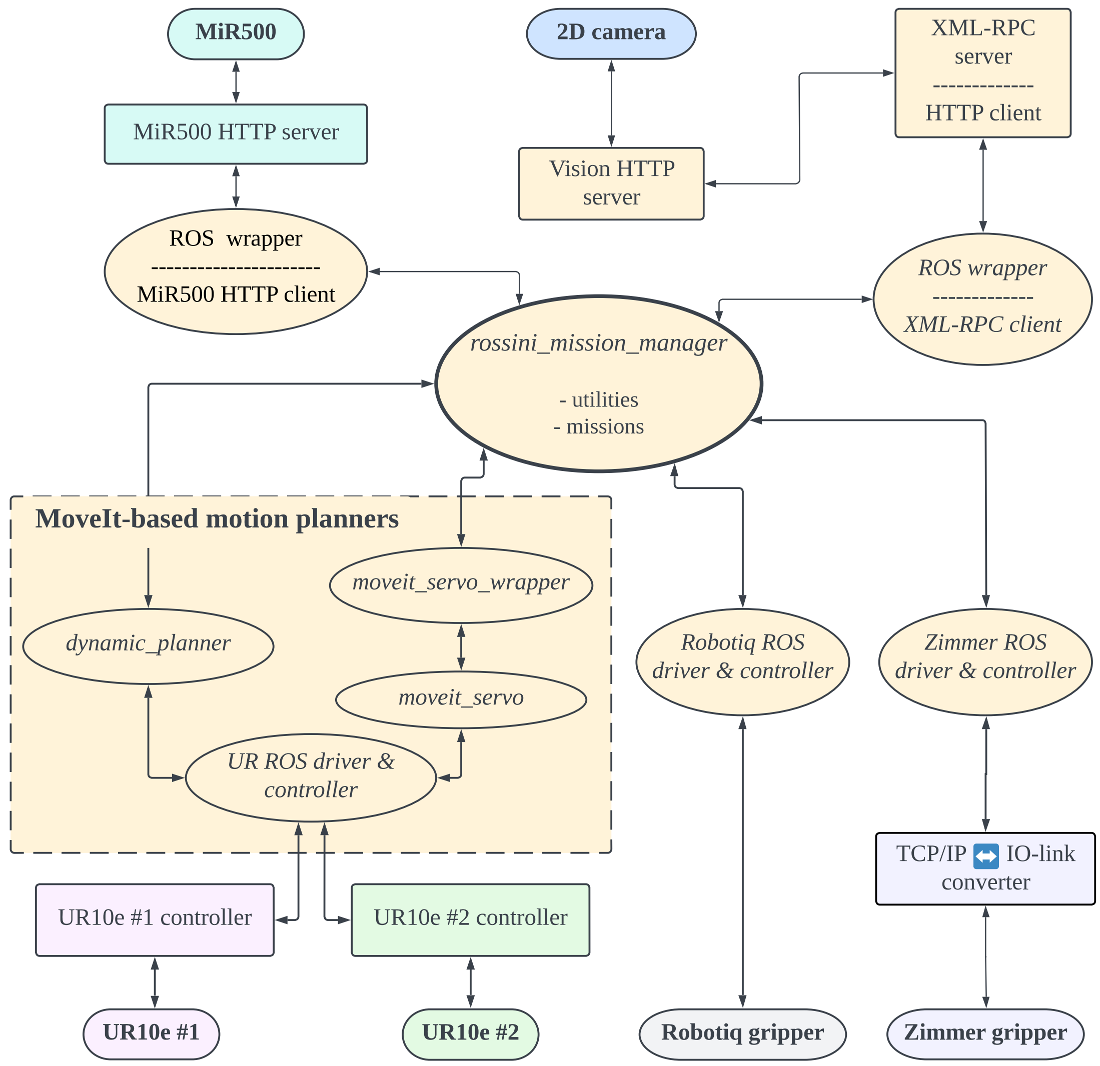


Figure 2.20: Basic functional software architecture for the ROSSINI AMR. The same color indicates that elements are deployed onto the same device (e.g. yellow = AMR workstation). Ellipses are ROS-related applications; rectangles are ROS-independent software.

sion queue thanks to an HTTP client that communicates with the MiR500 HTTP server launched by the proprietary software at AGV boot up.

As far as *ROS nodes* and *packages* strictly related to the ROSSINI project are concerned, three elements are worth mentioning: the task scheduler, the task generator and the mission manager. The first two runs on the ROSSINI workstation, while the last one is directly executed on the AMR to reduce latency in the communication with the motion planners and the UR drivers.

The *task scheduler* [42], developed by a research team of the University of Modena and Reggio Emilia (official partner of the ROSSINI consortium), is in charge of assigning machine-tending tasks to either a human agent or a robotic agent, e.g. the AMR. The tasks are dynamically scheduled based on execution constraints, the variability and job quality for the human in each task, and the real-time monitoring of the activities of both humans and robots involved. Tasks assigned to the AMR are sent to the mission manager, whereas tasks assigned to the human operator are sent through a suitable node to a web server accessed by the mobile app on the operator's phone.

The *task generator* is a simple node that mimics requests coming from one or more automatic machines. It emulates events such as the need for a replacement of one of the raw-material reels or a fault in the box-forming zone, which requires manual intervention. The requests are generated according to realistic processing times for material consumption and the probability of fault rising, obtained from the analysis of field test data and user experience. For the sake of demonstration, the time can be accelerated by a suitable factor so as to produce many types of requests in a reasonable time.

Lastly, the *mission manager* is based on a *ROS action server* which accepts a few types of missions and executes them in sequence. The missions implemented for the demonstration are the change of left[12] filter-paper reel and the change of left tag reel. The mission manager also provides feedback to the task scheduler about the outcome of the mission, as described in Sec. 3.6.

2.3.2 SENECA architecture and resources

In the SENECA project, the RS^4 system was removed, so all related packages and nodes are no longer needed. Likewise, we are not interested in task scheduling for this use case; therefore, the scanning mission is directly launched without the intermediate job of a task scheduler and a mission manager.

Regarding the software, only two new items were needed in the SENECA use case, whose principles are discussed in detail in Sec. 4. The first one generates the best path to scan the maximum amount of bin surface with the minimum effort, starting from the CAD model. A second one trains a model able to classify an input image framing a portion of the surface and tell whether it is clean or dirty. Both items must then be integrated into robot missions according to the *ROS architecture*.

The scanning path generation is realized mainly using three open-source libraries for Python, which is the language used for these implementations: *Open3D*[43], *PyMesh* [44] and *ACO* [45]. The classifier, instead, is implemented resorting to the well-known *PyTorch* [46] Python library.

[12] In the machine, there are two columns of mandrels to host, from top to bottom: a filter-paper reel, a tag reel and an outer-envelope reel. "Left", in this case, is referred to the left column.

Chapter 3

ROSSINI use case: automatic reel change

The IMA use case was selected between the three industrial applications by the EU commission in charge of evaluating the ROSSINI project because of its greater complexity and impact.

In this chapter, we describe in detail the procedure and algorithms involved in the use case. While the overall process is strongly inspired by the MaXima workflow (described in Sec. 2.1), the first (mainly technical) original contribution can be found in the new software implementation based on a paradigm that exploits ROS as the underlying architecture. In fact, even if the single actions that compose the tasks were almost the same, the different characteristics of the new AMR required a completely revised implementation to achieve the expected level of robustness and flexibility. All vision-related system pieces are also original integrations and adaptations of past works and computer-vision techniques that are suitably arranged for this specific industrial context to attain reliable and repeatable outcomes.

A brief summary of this use case is reported as the validation step in a real-life scenario of the ROSSINI generic framework described in our work-in-progress article [47], not yet published. We dedicated, instead, a full journal article [48] to the vision-based reel-picking process described in Sec. 3.3, given its impact in similar industrial scenarios.

3.1 Overview

Similarly to the MaXima use case outlined in Sec. 2.1, the objective consisted of the autonomous change of raw material to an automatic packaging machine by means of an AMR. Due to time constraints and limited added value to the ROSSINI demonstrator, the operation was restricted to the reels, thus neglecting the handling of cardboard blanks. Moreover, given the maximum payload of a single arm of 10 kg (including the weight of the tool), it was not possible to lift without dual-arm cooperation an outer-envelope reel, which weighed 10 kg by itself. Despite being an interesting topic, it was outside the scope of the project, and therefore, only filter-paper and tag-paper reels were targeted for the task. Nevertheless, the outer-envelope reel is taken back into account for the sake of completeness when discussing the reel-pose estimation phase in Sec. 3.3.1. In the following, we denote as a *mission* the macroscopic procedure that can be executed either by the AMR or by a human operator, such as the change of a paper

ID	Name	w_h	t_h [s]	w_r	t_r [s]
0	Micro-stoppage	0.4	120	9999	9999
1	Change Left Filter-Paper Reel	0.6	120	0.6	120
2	Change Left Tag-Paper Reel	0.4	120	0.4	120
3	Change Left Outer Envelop Reel	0.8	120	9999	9999
4	Change Right Filter-Paper Reel	0.6	120	9999	9999
5	Change Right Tag-Paper Reel	0.4	120	9999	9999
6	Change Right Outer Envelop Reel	0.8	120	9999	9999
7	Mission For Operator	0.1	100	9999	9999
8	Mission Assistance	0.1	100	9999	9999
9	System Shutdown	0.1	0.1	9999	9999

Table 3.1: IMA use-case tasks. w stands for *intrinsic cost*, t for *time*, h for *human* and r for *robot*.

reel or the handling of a system fault.

In the demo, a simple fake task generator running on the ROSSINI workstation is manually launched to emulate the requests that may come from one or more automatic machines. With reference to Sec. 2.3.1, the tasks accounted for by the task scheduler and their relative costs and expected execution times are listed in Table 3.1. Missions with ID ≥ 7 are artificial tasks that are triggered by the task manager either according to some internal logic or due to an anomaly that occurred and is raised by the AMR while performing an action."Mission for operator" (ID=7), in particular, is used to send through the mobile app a human operator to either check if the UR10e #2 arm mounts the right set of fingers or to complete the manual splicing operation[1] required after a successful reel loading. Setting a high cost for the robot practically prevents the task manager from assigning that task to the AMR. As a consequence, the only tasks that the AMR may potentially handle are the ones with ID equal to 1 or 2, depending on other information known to the scheduler, such as the availability of a human, the pending tasks and the tasks in progress. More details can be found in [42].

Upon reception of either of the two tasks, regardless of the reel type, the AMR leaves the charging station and starts approaching the wagon, where the reels are stocked in piles as described in Sec. 2.1.2. During this motion, the arms are retracted and blocked, and the safety is managed by the AGV only. If an obstacle is encountered along the path, the navigation system will try to re-plan the trajectory to reach the wagon, unless a solution cannot be found after 10 attempts or until the obstacle moves away. Failure management is addressed in Sec. 3.6 where fall-back procedures are outlined.

After docking the AMR to the wagon location, thanks to the VL-shaped marker, the reel-picking phase, described in detail in Sec. 3.3, begins. This phase includes a vision-based procedure for reel-pose estimation [48]. During the execution of this sub-task, the safety management is passed to the RS^4 system, which continuously monitors the scene to ensure that if a human approaches, the cobot first slows down (reduced mode, e.g. SSM) and immediately halts when contact is detected (e.g. SMS). In the latter case, the cobot stays paused until the human moves away and resumes the operation thereafter.

[1] The manual splicing job consists in unwinding the first strip of the reel and passing it through rollers and clamps. When finished, the automatic machine is able to automatically splice the strips from two different reels of the same type, so that the production is never interrupted.

Once the reel is stored on board, the AMR moves to the next work location, which is in front of the automatic machine. During the AGV motion, the safety is again managed by the MiR500 navigation system until docking, when it moves back to the RS^4 system.

After docking to the C24-E, the core of the consumed reel must be removed before loading the new reel on the same mandrel. The two operations are described in detail respectively in Sec. 3.4 and 3.5.

Upon completion, the AMR moves back to the charging station and awaits new missions, unless a new one is already pending in the queue and the battery charge is sufficient. In that case, the new operation starts immediately without stopping by the charging station.

Sec. 3.2 describes the preliminary calibration steps required to make the AMR work, whereas results of a few demo cycles are finally discussed in Sec. 3.7.

3.2 Calibration

A fundamental step before the ROSSINI AMR is fully operative is the system calibration. This operation includes:

1. **camera calibration**: estimation of the intrinsic parameters of the selected camera;

2. **eye-on-hand calibration**: estimation of the relative position of the camera with respect to the end-effector of the manipulator onto which is rigidly installed;

3. **robot-to-robot calibration**: estimation of the relative position of the two manipulators with respect to the mobile platform and, consequentially, to each other.

All these items are necessary to convert the information given by any vision-based algorithm (typically in the image space) into robot coordinates that can be used by the path planner and, consequentially, by the UR controllers.

3.2.1 Camera and eye-on-hand calibration

Exploiting the fact that the information needed for both camera and eye-on-hand calibration are the same, the procedure for the estimation of these parameters are merged into one.

To estimate the intrinsic parameters, a set of images framing a known pattern from a different perspective is necessary. The pattern can be of any sort and geometry, as long as it is easily and quickly detectable within the image. For this reason, a chessboard-like pattern is typically preferred because of its high contrast, which makes the corners, and therefore the whole pattern, evident at a very fine level.

In our case, a ChArUco board was selected (Fig. 3.1). This specific pattern includes a standard chessboard where each white box containes a specific ArUco marker[2] (ChArUco

[2] "An ArUco marker is a synthetic square marker composed by a wide black border and an inner binary matrix which determines its identifier (id). The black border facilitates its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. For instance, a marker size of 4x4 is composed by 16 bits.". Definition taken by *OpenCV* documentation.

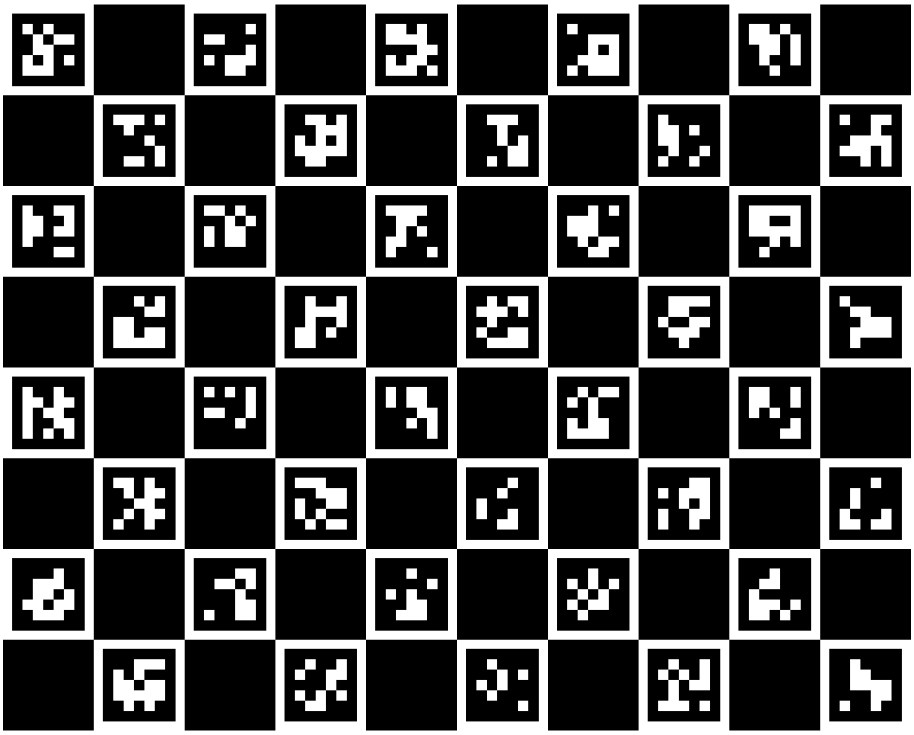


Figure 3.1: The calibration pattern: a 10x8 5bit ChArUco board.

= Chessboard + ArUco). This helps fit the whole pattern even when it is partially occluded or out of frame, thanks to the knowledge of the ArUco series, which allows obtaining the missing elements of the board by interpolation.

Hence, for each sample (i.e., image), a set of 2D coordinates expressed in pixels is obtained, where each pair represents an inner corner of the chessboard. The order of the coordinates is given by the ArUco series (Fig. 3.2). By knowing the exact size of the pattern in millimetres, it is possible to perform an optimization that fits the projection model that best matches each 3D corner expressed in the camera frame to the corresponding pixel in the 2D image space.

The projection model of a pinhole camera, illustrated in App. A, includes both the *intrinsic* characteristics of the sensor, that is, the physics of the lens of the camera, such as focal length and distortion, and the so-called *extrinsics* or *extrinsic parameters.* In general, the extrinsic parameters represent a rigid transformation between the camera frame and an arbitrary frame in which the 3D points are originally expressed before being projected onto the image plane. For the calibration process, this external system of reference is a frame whose origin is placed on the top left corner of the chessboard, with the x-axis oriented along the shorter dimension of the board, the y-axis along the longer dimension of the board and the z-axis pointing outside the board plane.

While the intrinsic parameters are static, time-invariant and bound to the physics of the sensor, the extrinsics are time-varying and depend on the relative position of the camera with respect to a suitably chosen "world" frame.

The outcome of standard camera calibration is, therefore, a parameter set including both the intrinsics and the extrinsics for each sample. The intrinsics determine the projection matrix and distortion model of the camera under investigation, whereas the extrinsics define the transformation between the camera frame and the chessboard-fixed frame for each picture shot (Fig. 3.3).

While the former is the actual desired results of the camera calibration, in our pro-

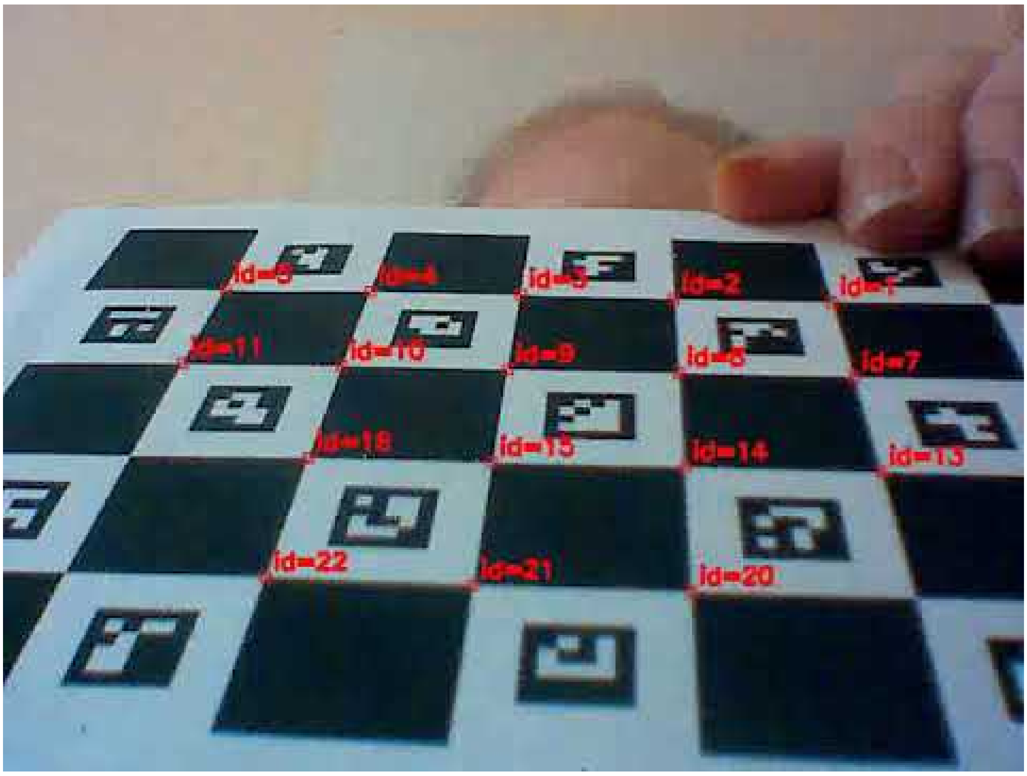


Figure 3.2: Example of corners detection. Source: OpenCV docs.

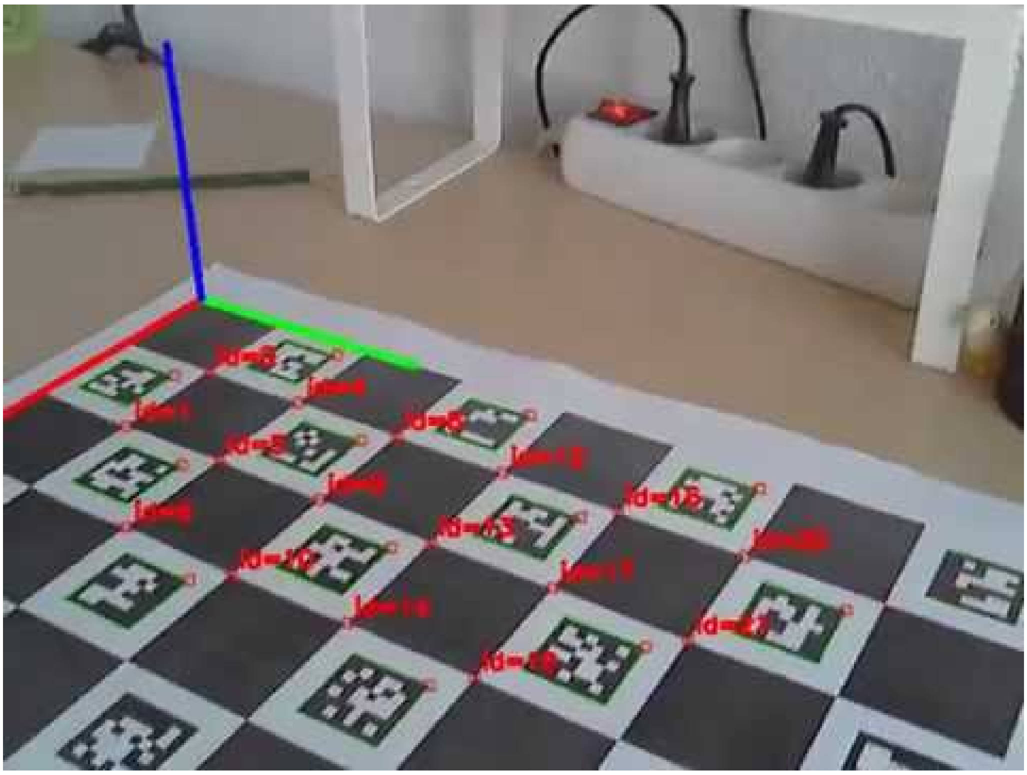


Figure 3.3: Board pose estimation for a single sample. Source: OpenCV docs.

cedure, we use the additional information coming from the extrinsics to set up the second calibration stage, which is the eye-on-hand calibration.

In the eye-on-hand scenario (see Fig. 3.4), the camera is mounted on the end-effector of a manipulator and therefore, it becomes relevant to express the camera frame with respect to the inertial frame of the robot itself, typically coincident with its fixed base. Usually, the pose of the end-effector is a piece of information obtained indirectly from the forward kinematics of a manipulator or more directly from the control system of the robot in the case of an industrial product. In both cases, the eye-on-hand calibration aims at finding that missing ring that is the static transformation between the end-effector and the camera frame. This transformation, in fact, does not change in time, assuming that the camera is rigidly installed on some support attached to the end-effector.

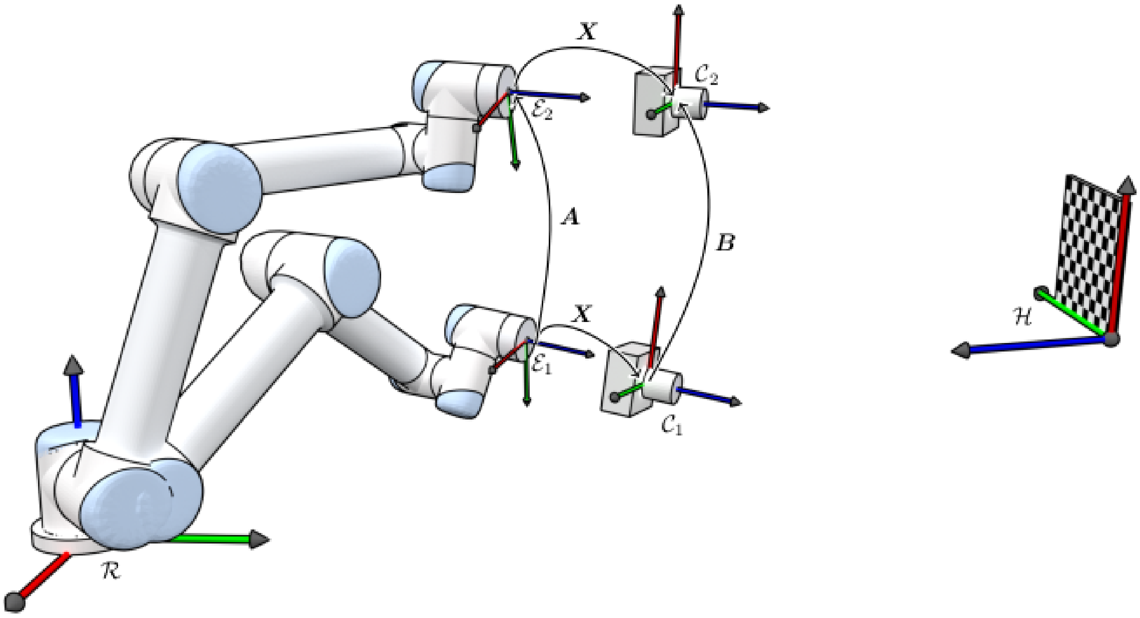


Figure 3.4: Eye-on-hand configuration. X is the static transformation between the end-effector and the camera frame, i.e. the result of the calibration process. Source: `https://www.torsteinmyhre.name/snippets/robcam_calibration.html`

The procedure employed here to estimate this transformation is based on two correlated data sets. On the one hand, there is the collection of extrinsics for each sample, that is, the transformation between the camera frame and the chessboard for each framing pose taken by the camera during the previous calibration motion. On the other, at every shot, the current end-effector pose expressed in the inertial reference system (i.e., the robot base) is stored and linked to that sample. As a result, we obtain a pair of transformations for each sample.

To compute the target transformation, two options were considered:

1. a *Least-Square-based* geometric approach directly implemented taking advantage of the work by Tsai and Lenz [49], whose details we leave to the reader to examine if of interest;
2. a *Neural-Network-based* approach, which was implemented for the MaXima project, and has been simply adapted to accept the same kind of input data as the first one.

Results obtained by the two methods are similar in terms of numerical values[3], but the former is significantly faster and therefore is the preferred/suggested option.

[3] The NN approach is by nature based on a stochastic iterative procedure, which makes it non-deterministic, differently from the Least-Square approach which has an analytical solution.

From a qualitative point of view, the outcome of the calibration is acceptable, and the success of the applications that were built upon this transformation is considered a sufficient indicator of its efficacy.

The calibration procedure can be launched as a stand-alone mission in two forms:

- *full calibration*: it calibrates both the camera intrinsic parameters and camera-to-TCP transformation for the eye-on-hand configuration; this option must be selected whenever there is a modification on the camera lens or at first installation;

- *eye-on-hand calibration*: it re-uses the previous camera intrinsic parameters and recomputes camera-to-TCP transformation only; this option can be selected whenever only the position of the camera with respect to the TCP is changed.

Both routines run in under 1 minute and are fully autonomous, provided that the correct ChArUco board is placed, even without much accuracy, in the expected position on the top surface of the AMR, i.e. the steel plate, as shown in Fig. 3.5.

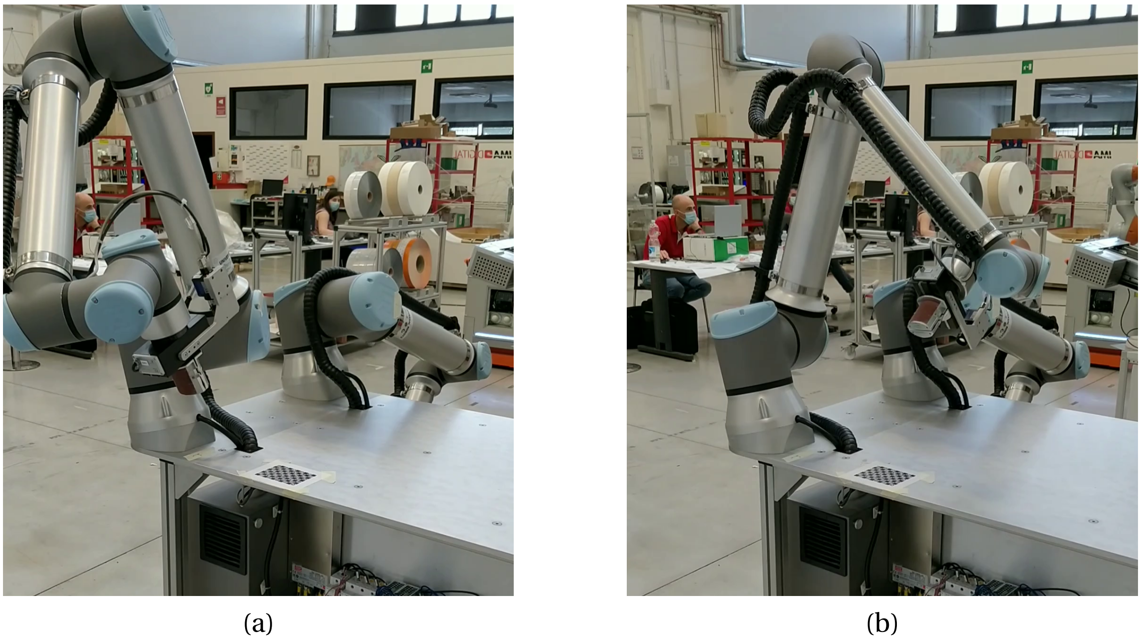

(a) (b)

Figure 3.5: Samples collection for the eye-on-hand camera calibration.

3.2.2 Robot-to-robot calibration

The robot-to-robot calibration is necessary whenever a piece of information is shared between the two manipulators, such as a common point of interest or the coordinates for a cooperative motion. This simple static transformation converts a point expressed in the reference frame of the UR #2, taken as the inertial frame, to the reference frame of the other UR #1. Thanks to the precise mechanical assembly of the whole system, the values of such transformation can be extracted directly from its CAD drawing without resorting to more sophisticated methods (see Fig. 3.6).

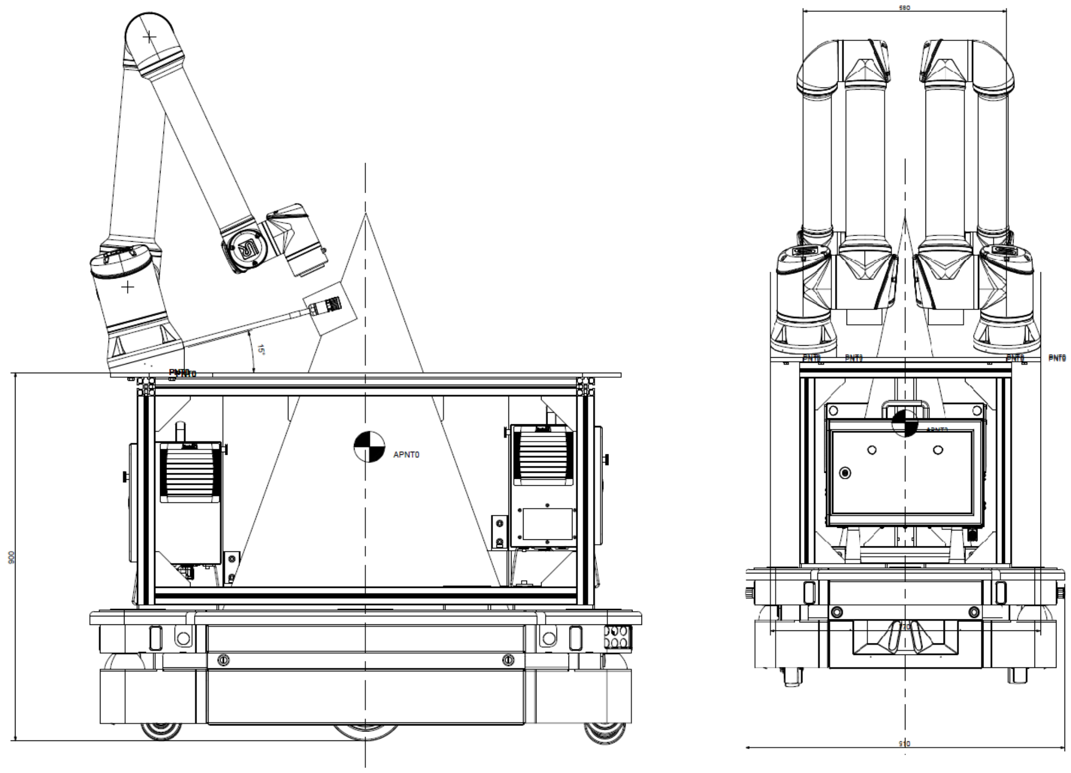


Figure 3.6: AMR CAD assembly.

3.3 Reel picking

Once the AMR docking to the wagon is complete, since arm #1 is not needed at this stage, it is moved to the so-called "away pose", which grants a larger manoeuvre space to arm #2.

As a second step, the wagon must be included in the virtual planning scene to avoid collision during the trajectory execution. However, its exact relative pose with respect to the arm #2 base, which is the planning reference system, is not known due to docking inaccuracy. Therefore, a simple box resembling the wagon maximum encumbrance with some extra margin is initially spawned in the scene in front of the AMR, as shown in Fig. 3.7.

Once the simplified obstacle is present, the cobot proceeds to the "marker scan pose" to frame and detect with millimetric accuracy a 5x4 ChArUco marker applied on the wagon's top shelf, as shown in Fig. 3.8.

Provided that the position of the tag is fixed and known with respect to the wagon, the box is replaced by a CAD model of the wagon in the actual measured position, as illustrated in Fig. 3.9. This CAD model at the beginning includes a full stack of all reels because, at this stage, it is not yet known how many are stored, and it is preferred to remain on the conservative side to avoid collisions.

Next, according to the targeted type of reel, the cobot moves to a suitable, predefined "reel framing pose", one per each type (e.g., filter paper on the top shelf, or tag paper, on the bottom shelf), as shown in Fig. 3.10.

At this stage, the wagon CAD model is replaced again with a similar version but with an empty stack on the target shelf (Fig. 3.11). This lets us add the actual reel at the front of the stack according to its measured pose estimated by the vision-based

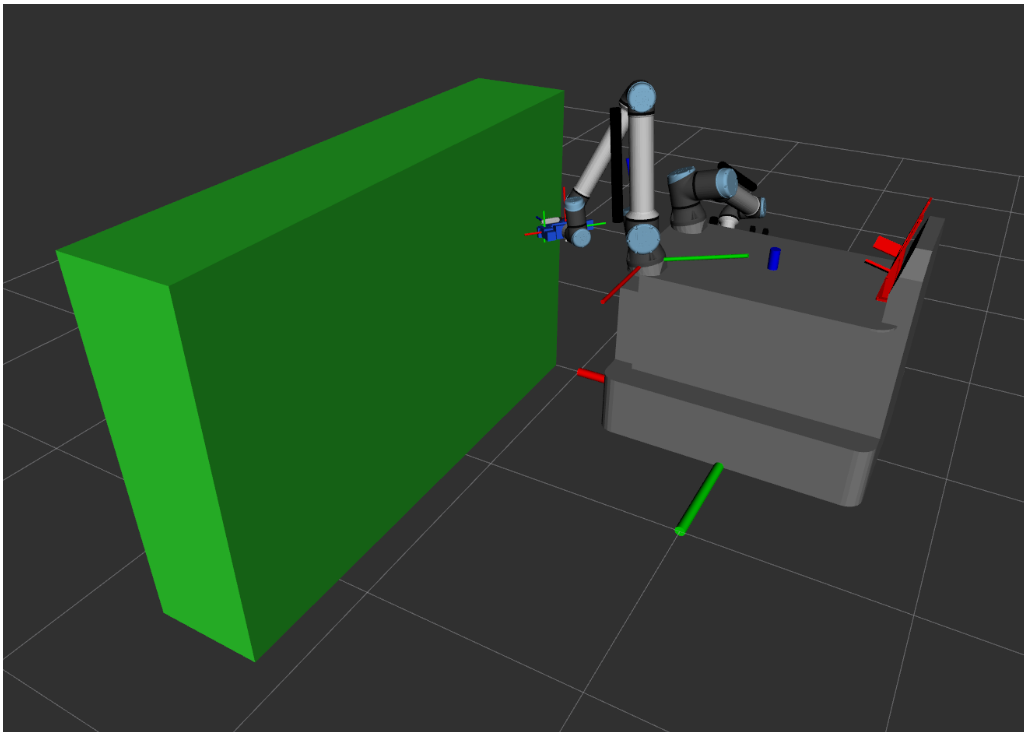


Figure 3.7: Spawning of a generic box resembling the wagon with a safety margin.

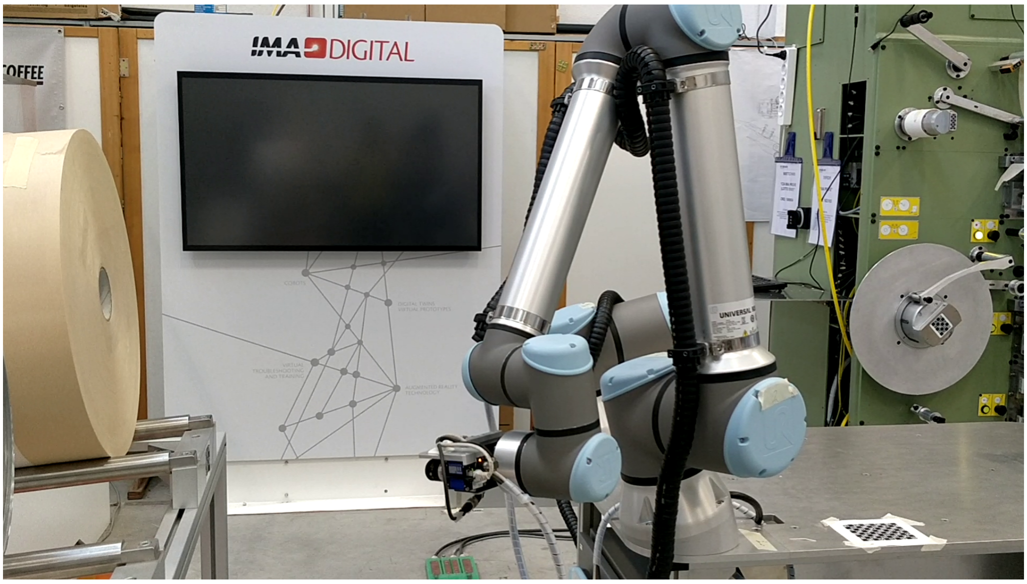


Figure 3.8: Marker scanning pose.

algorithm explained in Sec. 3.3.1.

3.3.1 Reel-pose estimation

3D circle-pose estimation from a 2D ellipse has been widely studied for its practical usefulness in several fields, such as eye direction recognition and localization of round-shaped objects. In [50], an analytical approach is given exploiting the geometrical properties of a projection camera. This idea relies on the evidence that according to the chosen perspective (i.e., reference camera frame), an oblique circular cone and an oblique elliptical cone provide the same projection of a circle onto an ellipse in 3D

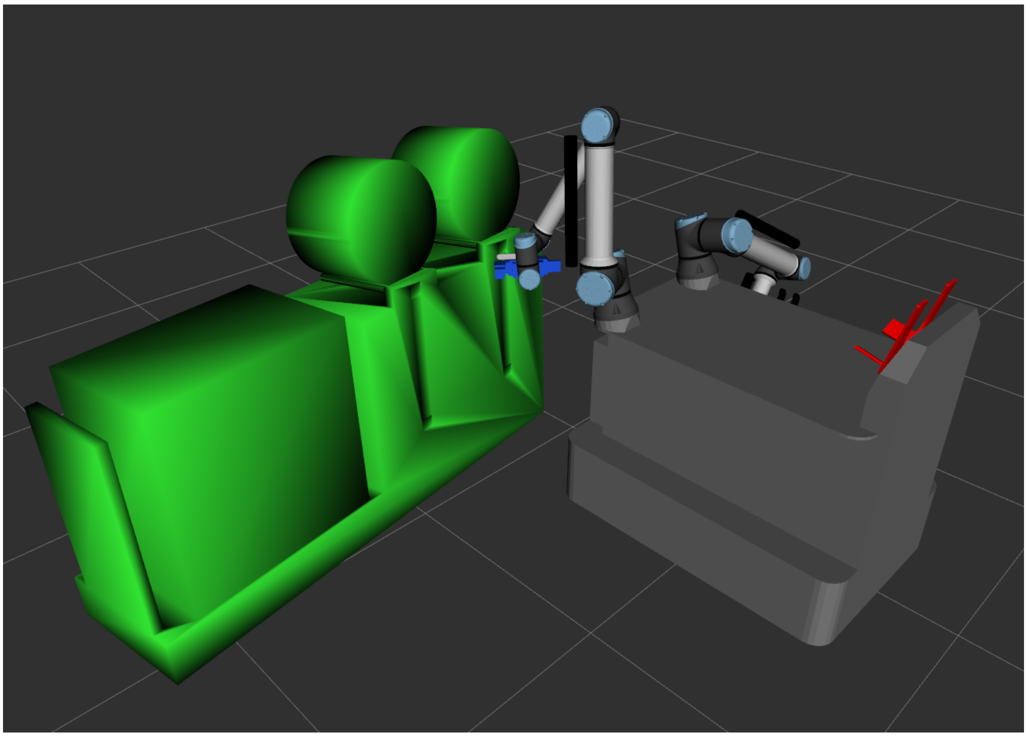


Figure 3.9: Spawning of the wagon after marker scan.

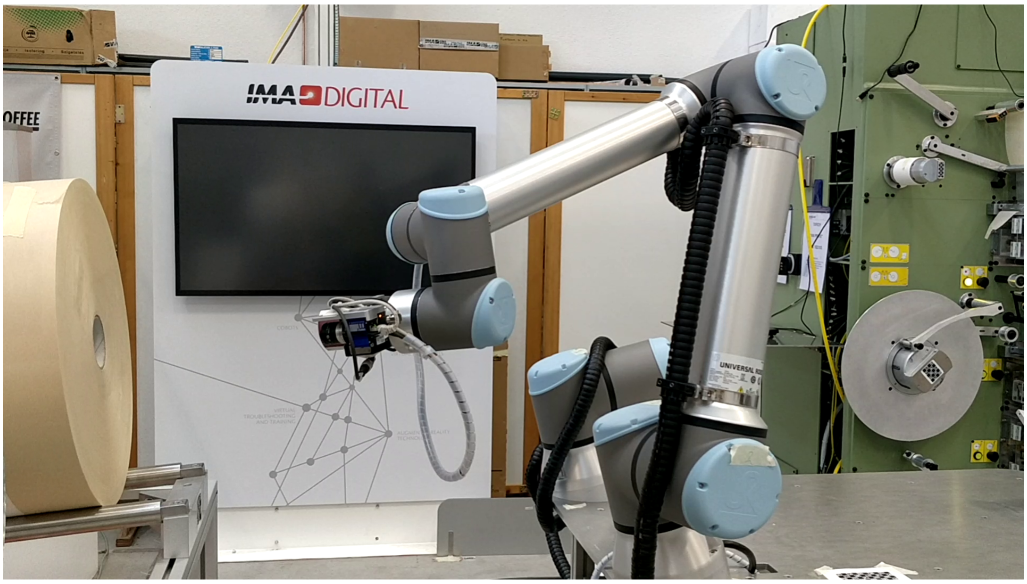
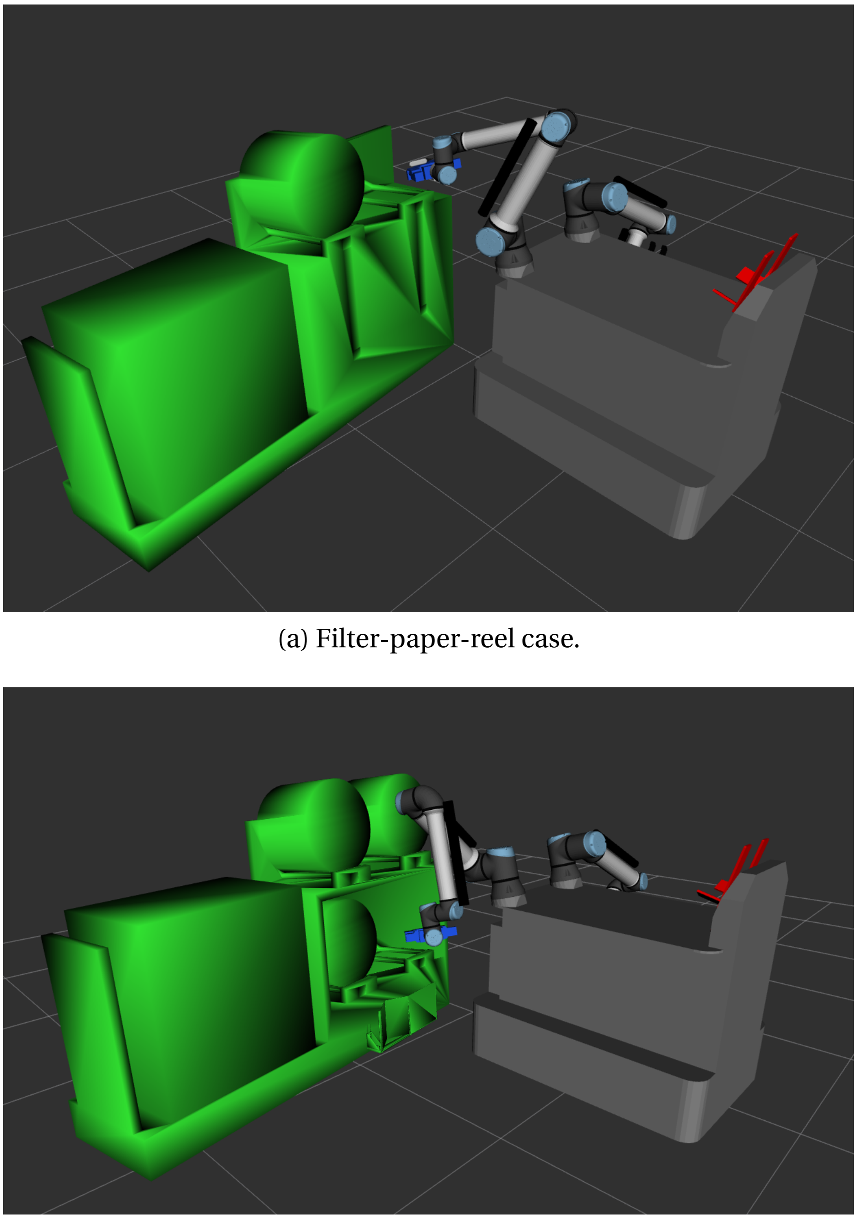


Figure 3.10: Framing pose of filter-paper-reel stack.

space and represent the same surface. Starting from a 2D ellipse detected on a single image, and given both the camera intrinsic parameters and the corresponding circumference radius in the real world, the outcome of this algorithm consists of a 3D point representing the centre of the circle expressed in the camera frame together with a unit vector normal to the plane the circle belongs to. It is important to notice, though, that, from a mathematical point of view, two valid solutions are always obtained as an outcome of this formulation. Intuitively, due to its symmetrical properties, it is impossible to distinguish between the two possible “directions” that can provide the same projection. A schematic view of this ambiguity is shown in Fig. 3.12.

To resolve this issue, we exploit the knowledge of the relative side-positioning of



(a) Filter-paper-reel case.

(b) Tag-paper-reel case.

Figure 3.11: Spawning of the wagon before reel-pose estimation.

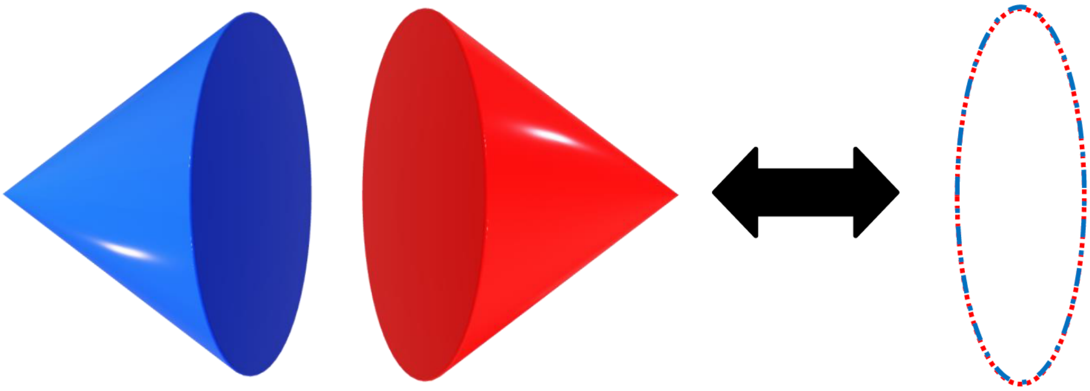


Figure 3.12: Circle projection ambiguity.

the camera with respect to the framed circle, that is, either right or left, which allows us to select the only realistic solution between the two.

The main assumption of the algorithm developed for ellipse detection is that a single reel core is always present within the image taken by the camera. This holds because once the wagon pose is available, a suitable framing configuration (relative to the wagon) can be manually determined once for all for each pile to guarantee that the core of each reel in the stack is visible inside the image (Fig. 3.10). In other words, the cone of view of the camera should include all the possible positions of the reel cores for the given pile. Alternatively, if the field of view of the camera is too narrow to frame all the cores from a single perspective, one could foresee a scanning motion that covers the whole length of the pile. In any case, the camera perspective should be chosen such that the circular section of the reel core appears as a slender ellipse in order to reduce the noise-to-signal ratio when back-projecting from 2D to 3D coordinates.

3.3.2 Reel-core detection

After extensive research about available algorithms and functions for ellipse detection and estimation, three methods were selected, adjusted, and rearranged in a pipeline that aims at covering most of the cases that can practically occur. The reason behind the choice of having multiple approaches instead of a single one is dictated by a trade-off between robustness and speed: while some methods may work well and fast for some specific scenarios, others might be, in general, more effective and robust, but take an unnecessarily long time. In a nutshell, we address the selected methods as follows:

1. *contour method*: relies on closed contours, which ideally should coincide with the target ellipse; it is very fast, and it works well in the presence of high contrast between the core and the surrounding wrapped material;

2. *RBrown method*: relies on edges including portions of the target ellipse; it is discreetly fast, and it works even in low-contrast conditions, but it is prone to false positives[4];

3. *Randomized Hough Transform*: relies on edges including the extremes of the target ellipse along the major axis direction; it is much slower than previous methods, but it is more robust even in low-contrast conditions.

In the use-case analyzed for this work, three types of reels are employed, whose main features are reported in Table 3.2.

Table 3.2: Reels features.

Reel Type	**ID**	**Core Diameter** (Inner/Outer)	**Thickness**
Paper	1	77/85 mm	100 mm
Tags	2	158/166 mm	30 mm
Outer Envelope	3	77/85 mm	100 mm

[4] In our context, a false positive is the detection of a wrong ellipse, which does not coincide with the true reel-core contour.

3.3.2.1 Ellipse detection by contour method

This method relies on the evidence that when the core is much darker than the surrounding material, the closed contour which encapsulates it is itself the target ellipse, corresponding to the external border of the cylindrical reel core. To extract the contour, the image needs to be converted to a binary form after the application of a threshold, which maps pixels darker than a certain value to black and the remaining ones to white. Due to different light conditions, this threshold is dynamically tuned by inspecting the histogram of the grayscale original image. Moreover, to get rid of noisy features like dots and black spots, the original image goes through several blurring phases until a proper contour is found. Nevertheless, due to imperfections on the real border, this contour can present small flaws or unexpected edges, which, however, are usually non-significant. Hence, a proper contour is defined according to the area of the surface it encapsulates and to the "goodness" of the ellipse fit through the least-square method, given by the residual error between the true contour points and the points of the estimate. Figure 3.13 depicts the data flow for the described method, whereas Figure 3.14 illustrates the main steps.

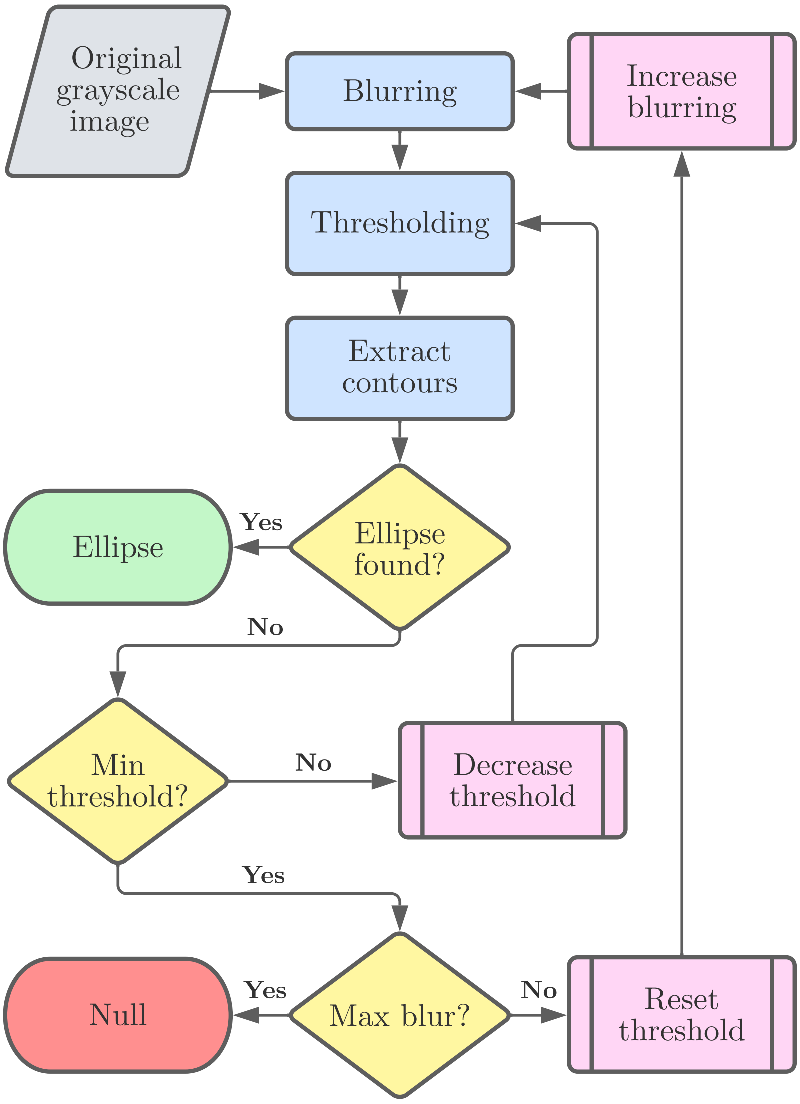
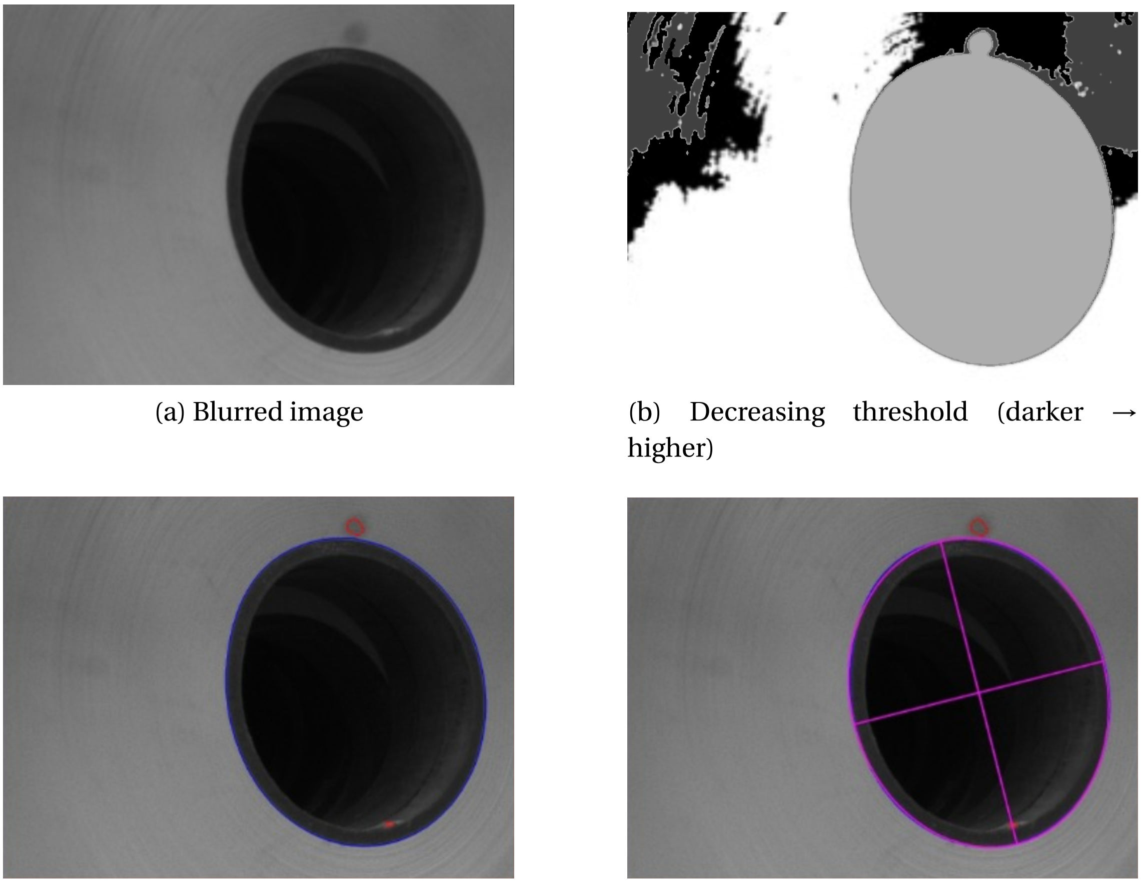


Figure 3.13: Contour method flowchart.



(a) Blurred image

(b) Decreasing threshold (darker → higher)

(c) Contours

(d) Detected ellipse

Figure 3.14: Contour method steps.

3.3.2.2 Ellipse detection by RBrown method

It is not always possible to extract a clean, closed contour from an image. Due to adverse light conditions, the contrast between the core and the surrounding material might be very low and lead to an impossible distinction between the two parts, sometimes even for the human eye. Employing edges instead of contours, we switch the focus of the algorithm to actively close a whole ellipse starting from a single solid portion of it. This method is also effective for those cases where the ellipse is not completely visible due to occlusions or for being partially out-of-frame rather than for low-contrast reasons. Nonetheless, it is prone to produce false positives whenever the portion under consideration has a small curvature (i.e. close to the extremes of the ellipse minor axis). On the contrary, it provides very good results when the arc includes the location of maximum curvature points, which correspond to those points close to the extremes of the major axis. This is related to the noise-to-signal ratio, which amplifies errors given by noisy edges when the curvature is low and vice-versa.

From a process point of view, the first step is the extraction of proper edges. Parameters of standard functions for edge detection are dynamically tuned so as to produce a minimum number of solid edges of a given minimum length (in pixels). Moreover, close-by edges are actively connected by binary closing operations[5] and skeletoniza-

[5] *Closing* is a mathematical morphology operation that consists in the succession of *dilation* and *erosion* of the input with the same structuring element. Closing, therefore, fills holes smaller than the structuring element.

tion[6].

For each edge in the set, two ellipse-fitting functions, respectively derived from the work of [51] and [52], are invoked and, therefore, up to two candidate ellipses are generated:

1. a linear Least-Square-ellipse fit by using the Bookstein constraint;

2. a non-linear (Gauss-Newton) least-square-ellipse fit based on an initial guess given by linear least-square-ellipse fit using trace constraint.

Both candidates for every single edge are validated through the same "goodness" check presented in Sec. 3.3.2.1.

The collection of candidates of all edges found is then optionally filtered using additional information provided by the user, such as the expected ranges for aspect ratio or major-axis length. A simple outlier removal based on centroids and aspect ratios is used to refine the collection. Finally, the smaller or larger ellipse among all candidates is taken according to the reel type (from experience, with reference to Table 3.2, for ID=1 or 2, the larger ellipse coincides with the external border of the reel core, while for ID=3 the smaller ellipse coincides with the more evident internal border). A schematic view of the mentioned process is displayed in Figure 3.15, whereas Figure 3.16 shows the main steps.

[6] The *topological skeleton* of a shape is a thin version of that shape that is equidistant to its boundaries.

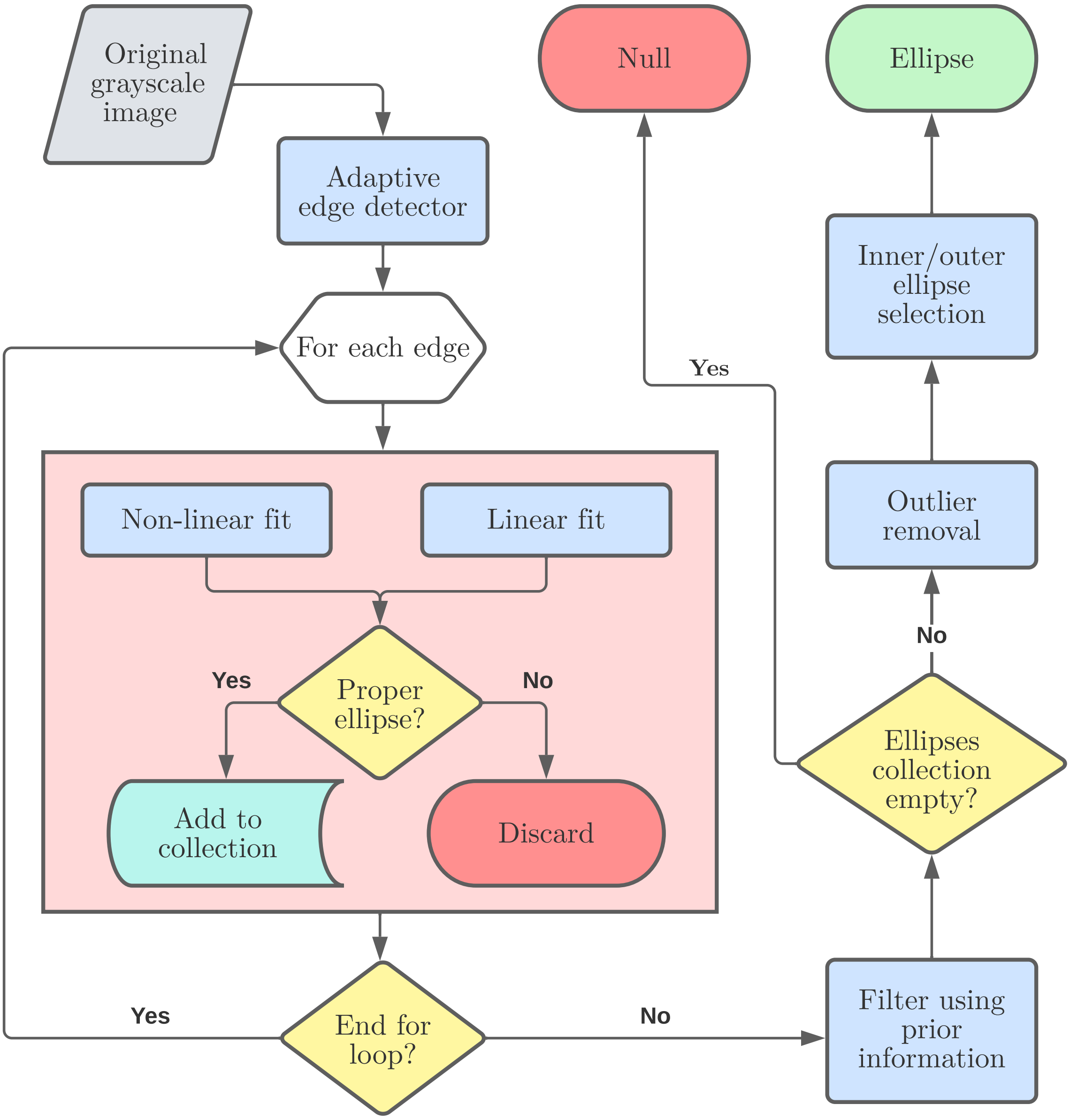


Figure 3.15: RBrown method flowchart.

3.3.2.3 Ellipse detection by randomized Hough transform

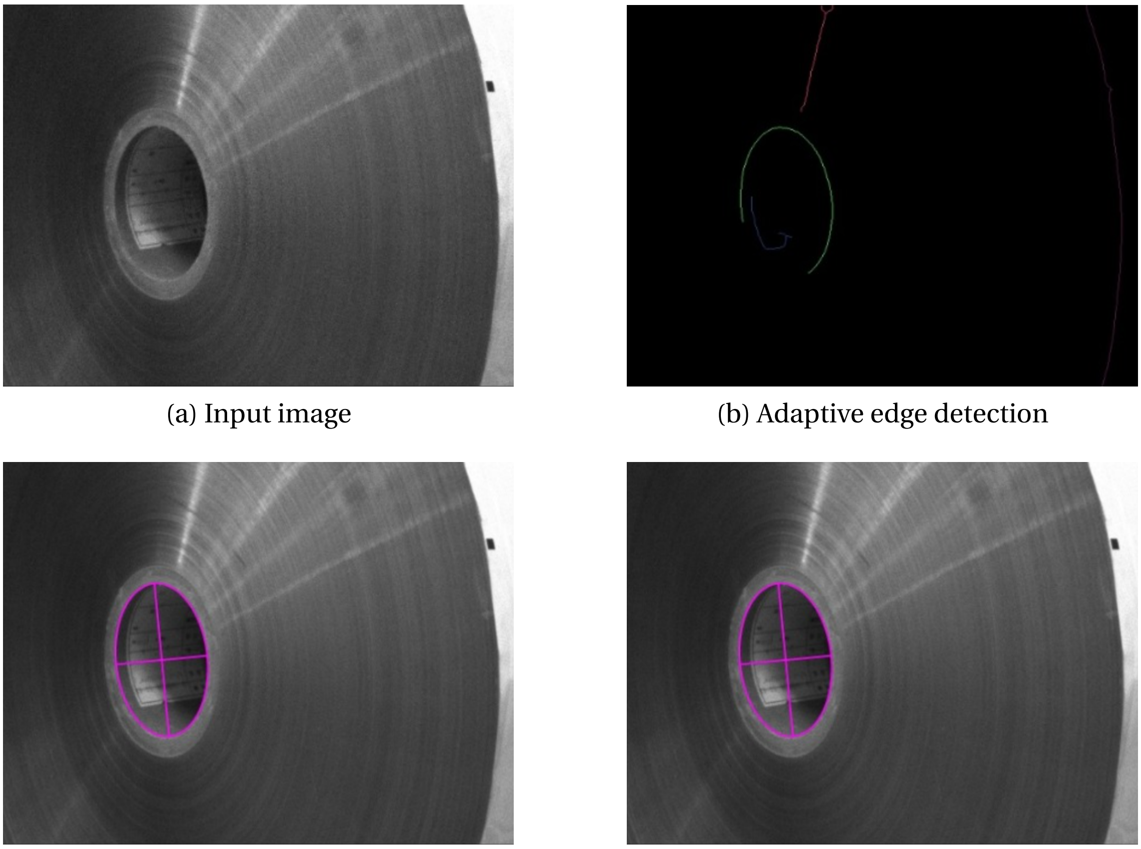
Like the RBrown method presented in Sec. 3.3.2.2, the Hough transform also relies on edges instead of contours, but, on the contrary, it uses all of them at once to find the best fit.

The algorithm is a direct implementation of [53] and [54], and it assumes that both extremes of the major axis are present within the edges. Being a loop-based algorithmic implementation, the key to improving the speed of this computationally expensive process are filters and randomization[7].

Once again, prior knowledge of the target ellipse drastically reduces the number of iterations and, therefore, is strongly recommended to exploit information such as the expected dimension and shape (i.e., aspect ratio). Moreover, in this context, a new filter based on tangent angles was developed and described in detail in App. C.

Briefly, each iteration picks a pair of two points among the edges and tries them as

[7] Random sub-sampling of original data.



(a) Input image

(b) Adaptive edge detection

(c) Ellipse fitting

(d) Detected ellipse

Figure 3.16: RBrown method steps.

extremes of the major axis. Moreover, because we work with solid edges and not sparse points, each picked pixel point belongs to a digital curve (i.e. a solid edge) and, therefore, can be assigned a tangent-angle value in the range (-90°:90°]. Exploiting these two facts, we can further restrict the selection of pairs to those points that share similar tangent angles, being de facto two opposite points in the ellipse.

A scheme of the data flow for this last method is shown in Figure 3.17, whereas Figure 3.18 illustrates its main steps.

3.3.2.4 Method selection policy

From previous consideration and experience, we outlined a selection flow for picking the best method according to the different characteristics of each reel. In particular, paper and tag reels (ID=1,2) show high contrast between the core and the surrounding wrapped material and, therefore, are suitable for the contour method, which is also the fastest of the three. RBrown and Hough transform are thus kept as successive fallback options in case the first method fails, being more robust but also slower. The outer envelope reel is highly reflective, and it is impossible to clearly extract a closed contour from a grey-scale image. For this reel type, the best performing method is the Hough transform, while RBrown is kept as a fallback option in those rare cases when it fails (for instance, when both major-axis extremes do not appear in the edges). Additionally, to speed up computation time, for Rbrown and Hough transform methods, a downsized version of the original image is used (resized to 480p), and several tests

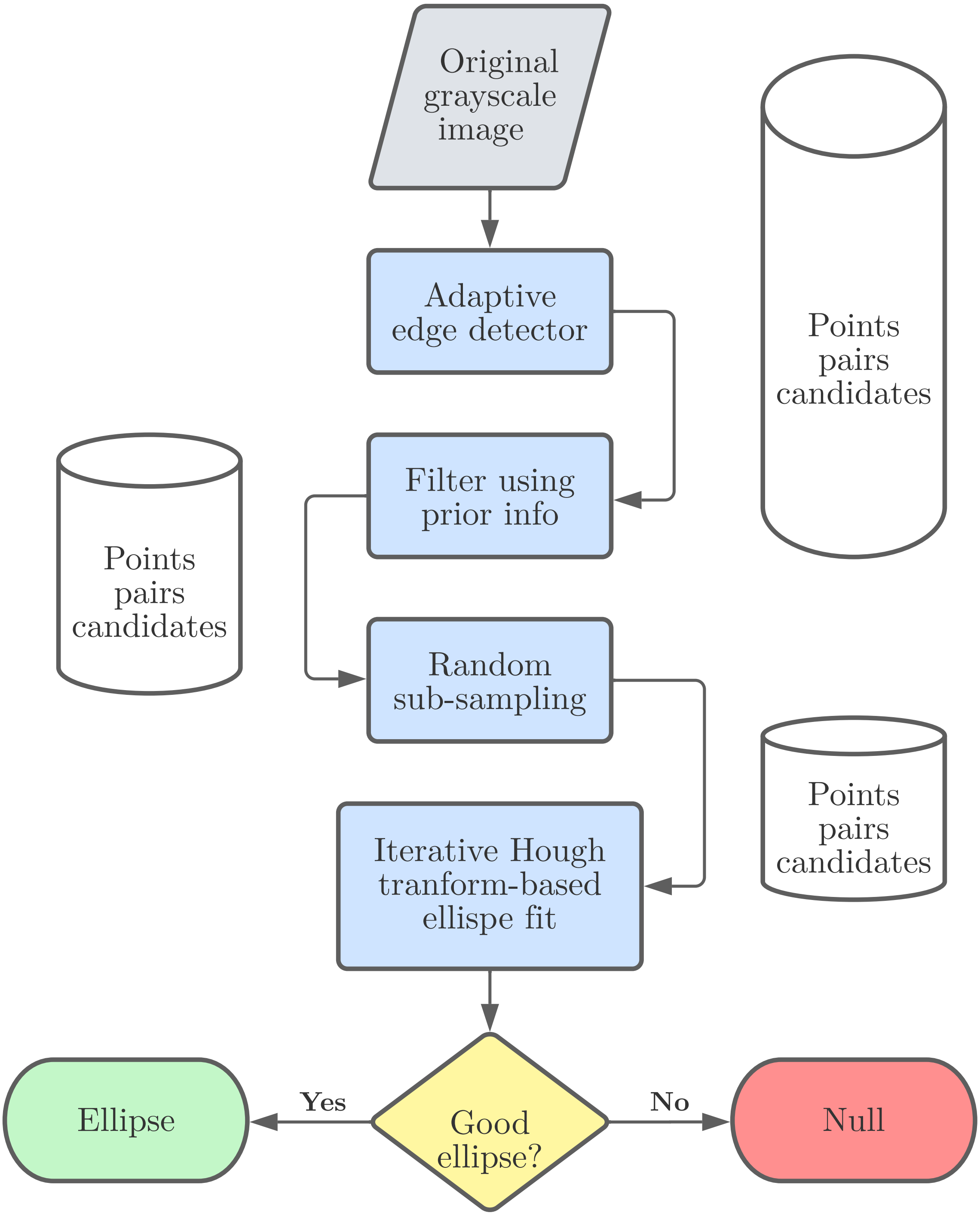
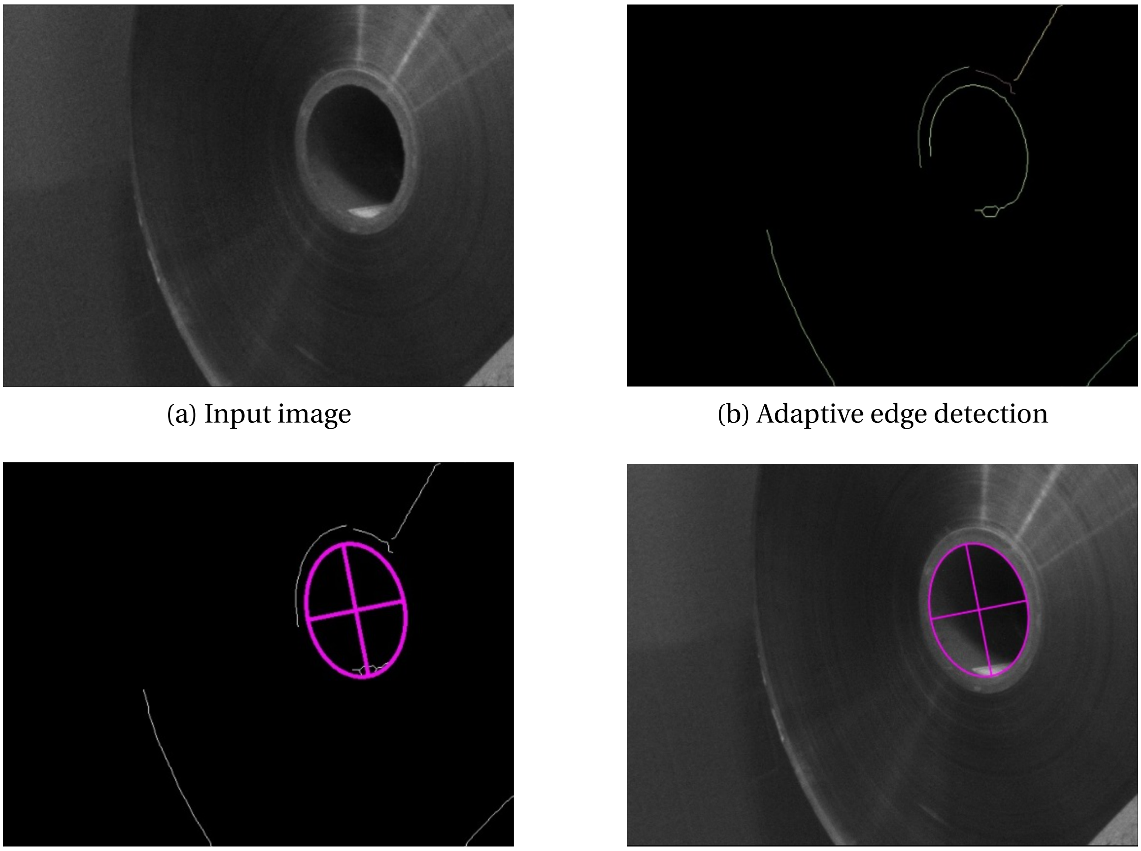


Figure 3.17: Hough transform method flowchart.

showed a negligible degradation of performance in terms of accuracy, sometimes even leading to improvements. Moreover, because Rbrown and Hough transform methods share the same starting phase, that is, edge detection, in case of failure, extracted edges are reused by the next method to save uptime. A schematic view of the selection policy is illustrated in Figure 3.19.

In the context of the ROSSINI use case, however, only filter-paper and tag-paper reels were actually considered, as already mentioned at the beginning of Ch. 3.



(a) Input image

(b) Adaptive edge detection

(c) Ellipse fitting

(d) Detected ellipse

Figure 3.18: Hough-transform method steps.

3.3.3 Reel grasping

Once the pose of the reel core is known, the 3D model of the reel is spawned in the scene and added to the stack, as shown in Fig. 3.20. To avoid collision with the rest of the pile when the reel is grabbed, a second reel is also spawned behind the one that will be picked up. Next, the robot moves its TCP[8] 10 cm in front of its centre (Fig. 3.21a), and, to insert the gripper fingers, moves linearly towards the reel in the z-direction of its reference frame (i.e. the longitudinal axis of the cylinder that represents the reel core) until contact (Fig. 3.21b). Because the thickness of the reel is known, given the predefined approach distance and the geometry of the gripper, it is possible to guess the expected length of the path up to the stop. If the estimation is not exact, due to the little clearance between the size of the gripper fingers and the diameter of the reel, the gripper will hit the reel before running the whole insertion distance, resulting in a failure. On the contrary, if a contact is not detected beyond the expected insertion length, an error is raised.

After the insertion, the gripper opens the jaws for inside grasping, lifts up the reel vertically (Fig. 3.21c), and a trajectory is dynamically computed to bring the reel close to the buffer aboard the AMR (Fig. 3.21d). The planning, in this case, is constrained to have the z-axis of the TCP with an azimuth angle above -15° with respect to the horizon. This avoids the risk of the reel slipping off when the TCP points too far down.

The depositing motion towards the buffer consists of a sequence of linear motions

[8] In this case, located at the tip of the gripper fingers.

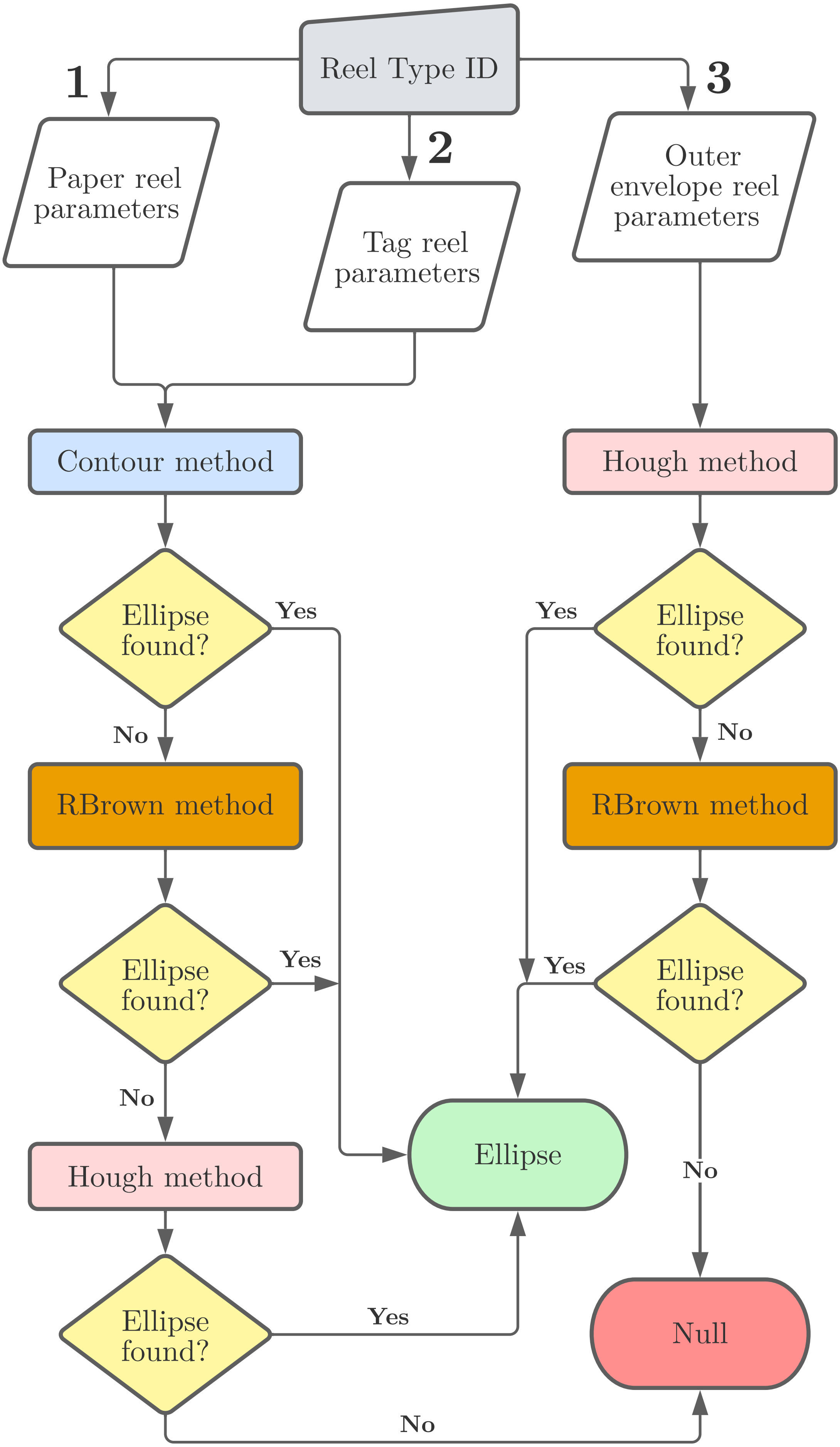
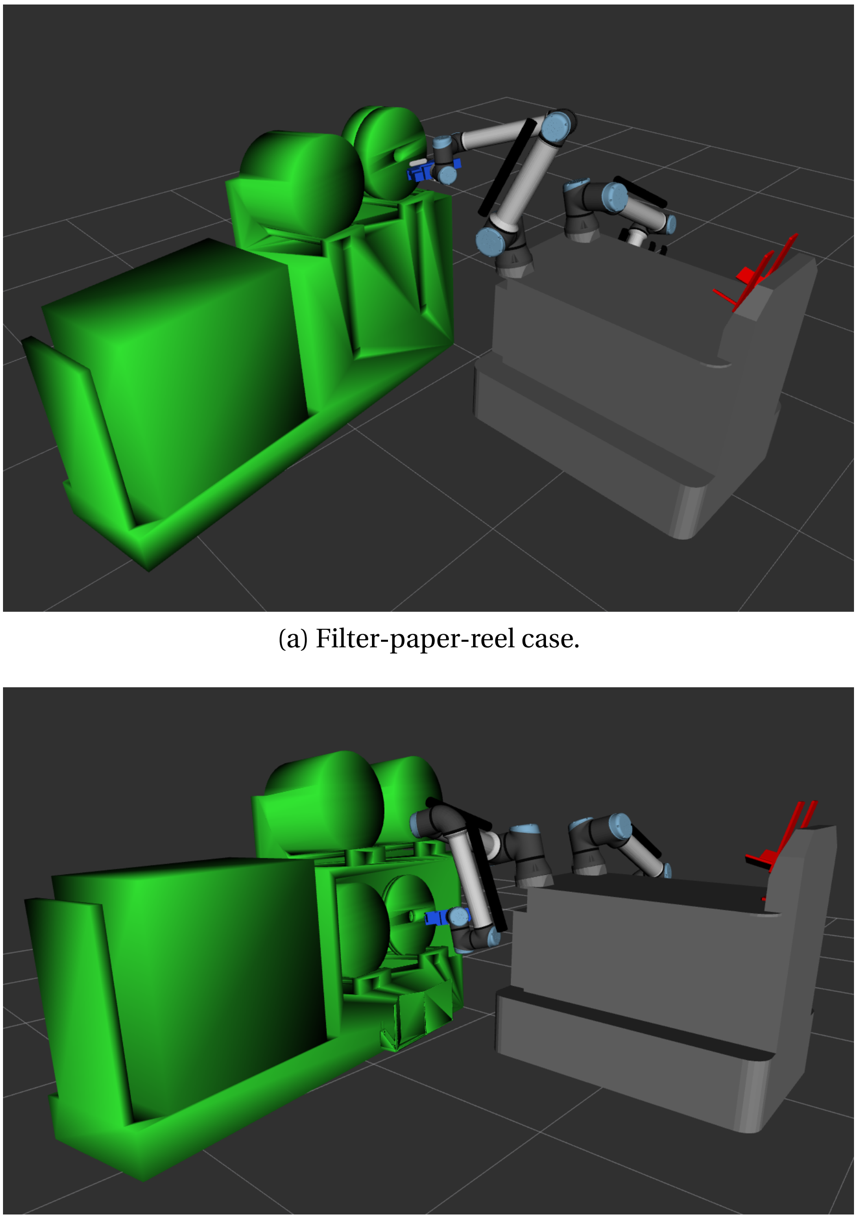


Figure 3.19: Method selection flowchart.



(a) Filter-paper-reel case.

(b) Tag-paper-reel case.

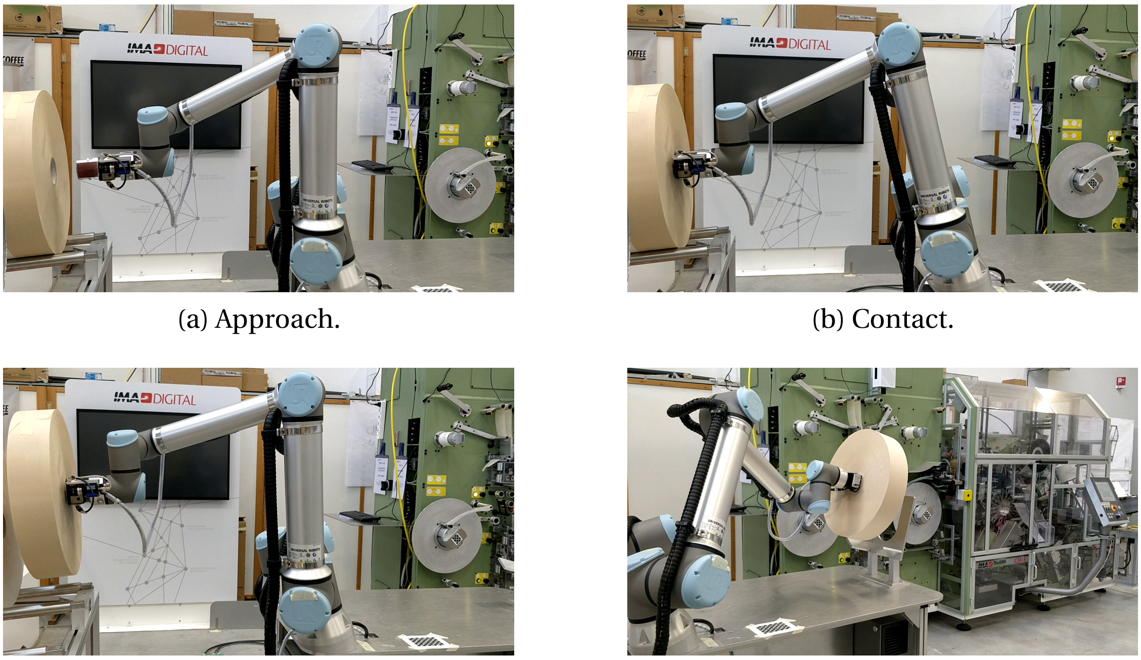
Figure 3.20: Virtual planning scene after reel-pose estimation.

to account for different diameters of the reel, whose true dimension depends on the amount of filter paper or tags wrapped around the core. When the reel is resting on the buffer, the gripper disengages, and the position is saved for later use.

Lastly, the gripper fingers are retracted by a backward linear motion (i.e. $-z$-direction in the TCP frame), and both arms move back to the original idle configuration.

3.4 Reel-core disposal

Both reel-core disposal and reel loading (Sec. 3.5) require the knowledge of the exact position of the mandrel where the target reel should be replaced. Similarly to reel picking, the presence of a ChArUco marker is exploited. Two tags are, in fact, applied to the tip of the mandrels: a 5x4 marker for the tag-paper-reel mandrel (Fig. 3.22a) and a 10x2 marker for the filter-paper-reel mandrel (Fig. 3.22b).



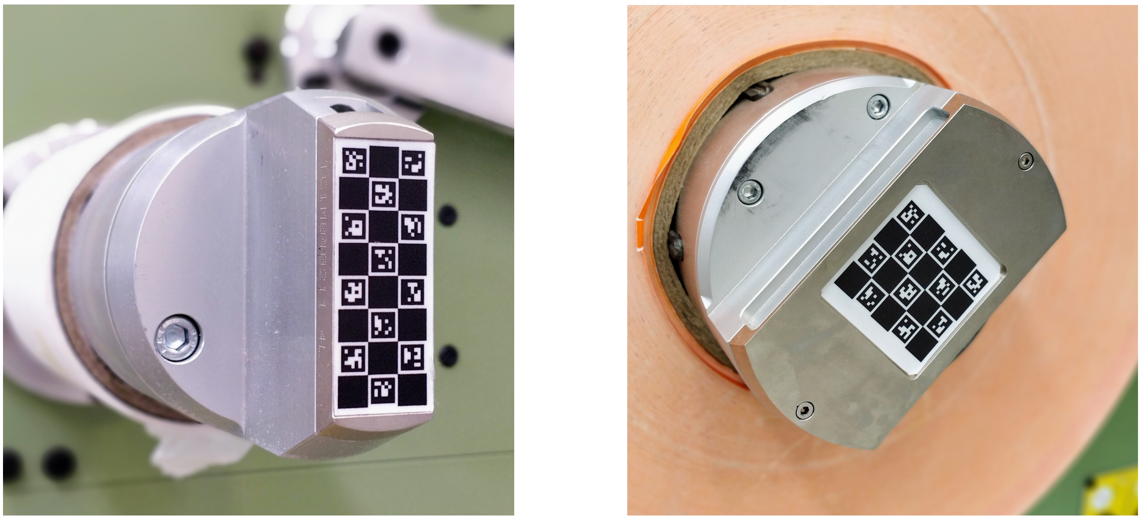
(a) Approach.

(b) Contact.

(c) Lift up.

(d) Buffer approach.

Figure 3.21: First steps of filter-paper-reel picking.



(a) Filter-paper-reel mandrel.

(b) Tag-paper-reel mandrel.

Figure 3.22: ChArUco markers on the C24-E automatic machine.

Hence, as a first step, a box resembling the dimension of the automatic machine with an extra margin is spawned in the planning scene after docking, and arm #2 moves to the "mandrel scan pose", as shown in Fig. 3.23 and 3.24. Differently from the wagon, this pose varies according to the reel type and, thus, the relative mandrel position in the C24-E.

Once the marker pose is known, a simplified version of the CAD model of the C24-E replaces the collision box in the planning scene, as illustrated in Fig. 3.25.

The next step is the only part of the reel change procedure that substantially changes depending on the reel type. In fact, arm #1 and adaptive grasping are employed for the filter-paper-reel core, whereas the same arm #2 in charge of loading the reel is also used to remove the tag-paper-reel core.

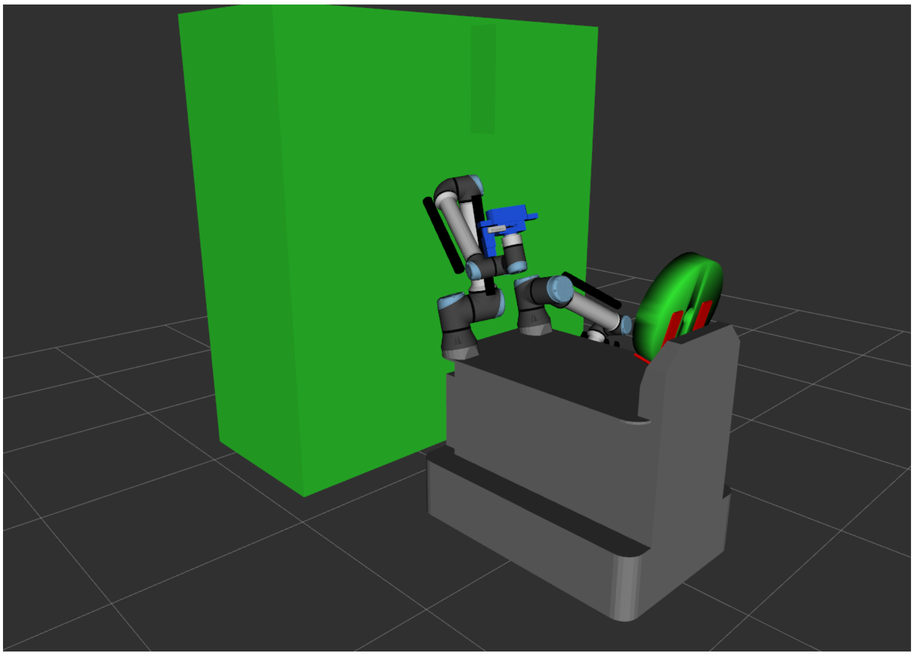


Figure 3.23: Virtual planning scene before marker scan.

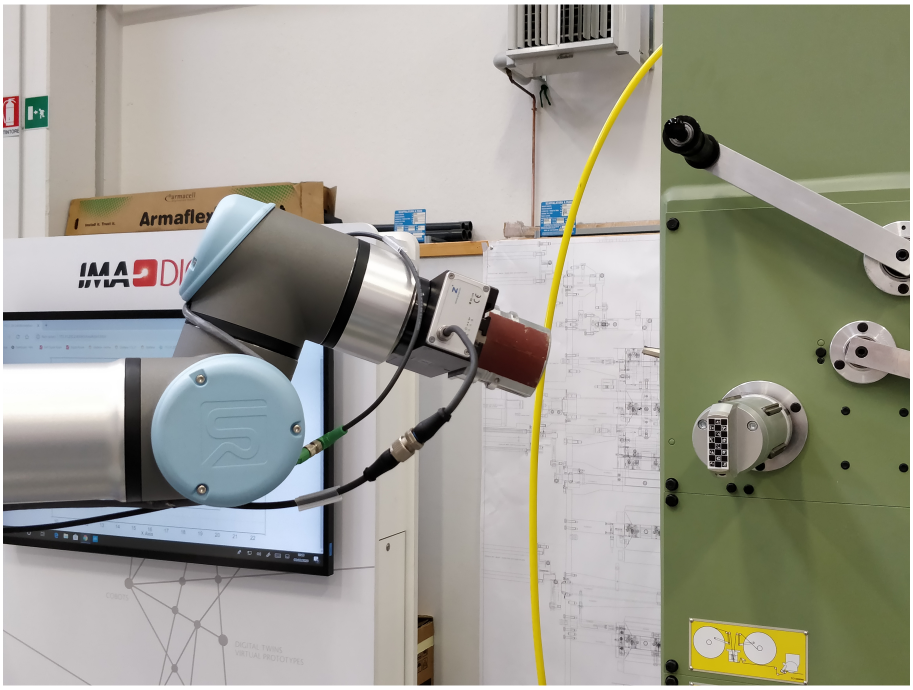


Figure 3.24: Filter-paper-reel mandrel scan.

3.4.1 Filter-paper-reel-core removal

The marker pose is measured with respect to the base frame of arm #2, which is the one equipped with the camera. This pose is then transformed into the base frame of arm #1 to dynamically compute a suitable pose that approaches the reel core from the side, as shown in Fig. 3.26a. The gripper then advances towards the core until contact and

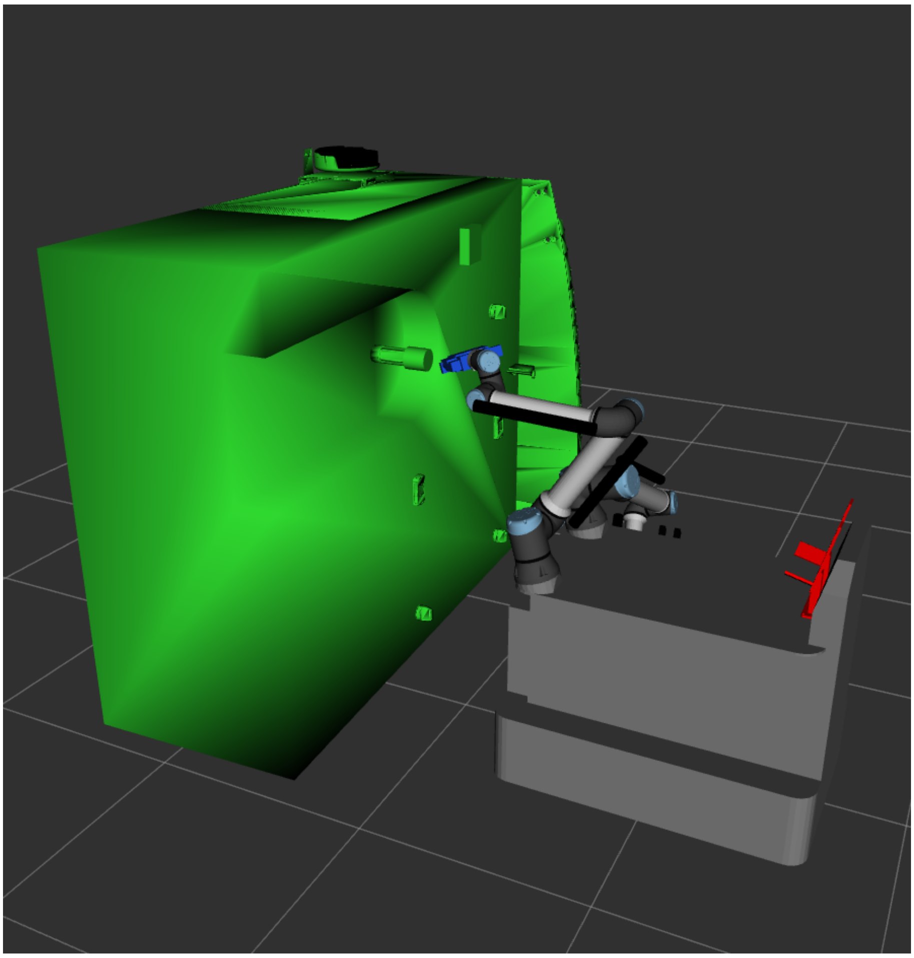
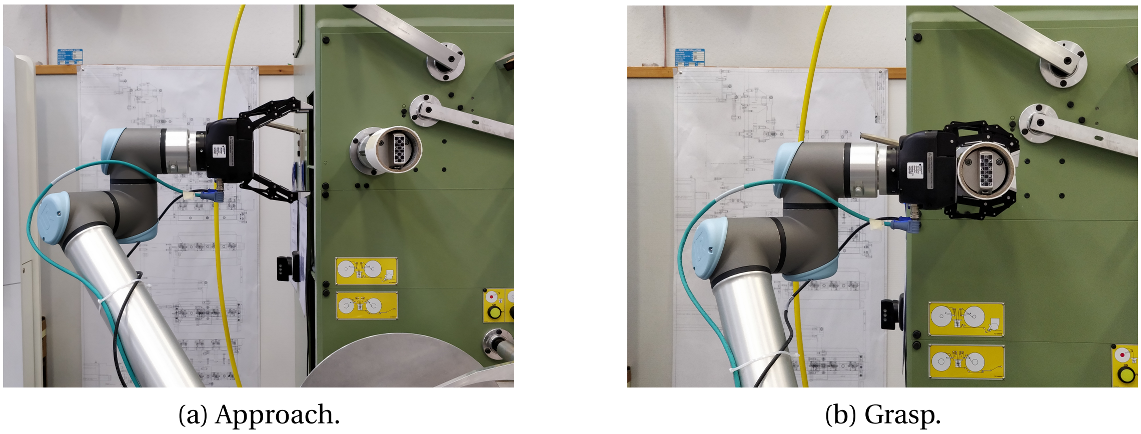


Figure 3.25: Virtual planning scene after the scanning of the marker on the filter-paper-mandrel.



(a) Approach.

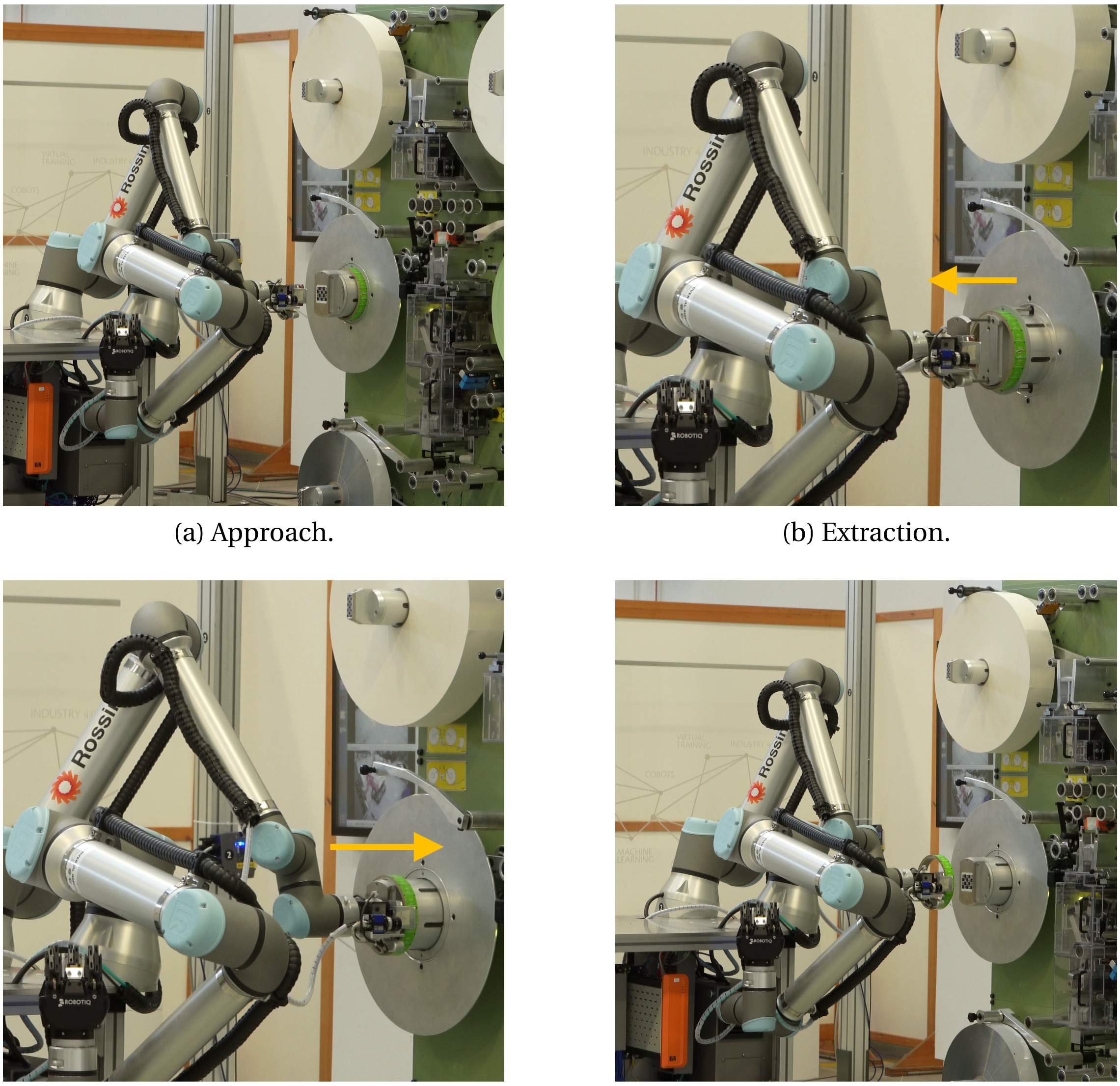
(b) Grasp.

Figure 3.26: Grasping of a filter-paper-reel core.

closes its fingers that adaptively grasp it, as illustrated in Fig. 3.26b. Afterwards, the end-effector moves sideways along the direction of the mandrel axis (which coincided with the z-axis of the detected marker) by a predefined length. Next, the manipulator brings the reel core above the platform plate, releases it and moves "away" to let more manoeuvre space for the other arm during the loading step, described in Sec. 3.5.

3.4.2 Tag-paper-reel-core removal

Since the diameter of the tag-paper-reel core exceeds the stroke that can be covered by the 3-finger adaptive gripper, its removal requires a different strategy. The mandrel that hosts the tag reel is equipped with a mobile sliding body. Cobot #2 first positions itself in front of the mandrel and opens the jaws by a small amount slightly smaller than the core internal diameter (Fig. 3.27a). It inserts the finger by a predefined length, grasps the mandrel "nose", and pulls the mobile body that, in turn, pushes the core out along the mandrel (Fig. 3.27b). The mobile body is pushed back by the cobot fingers to its original position (Fig. 3.27c), and the gripper can now grab the core from the inside and deposit it (Fig. 3.27d).



(a) Approach.

(b) Extraction.

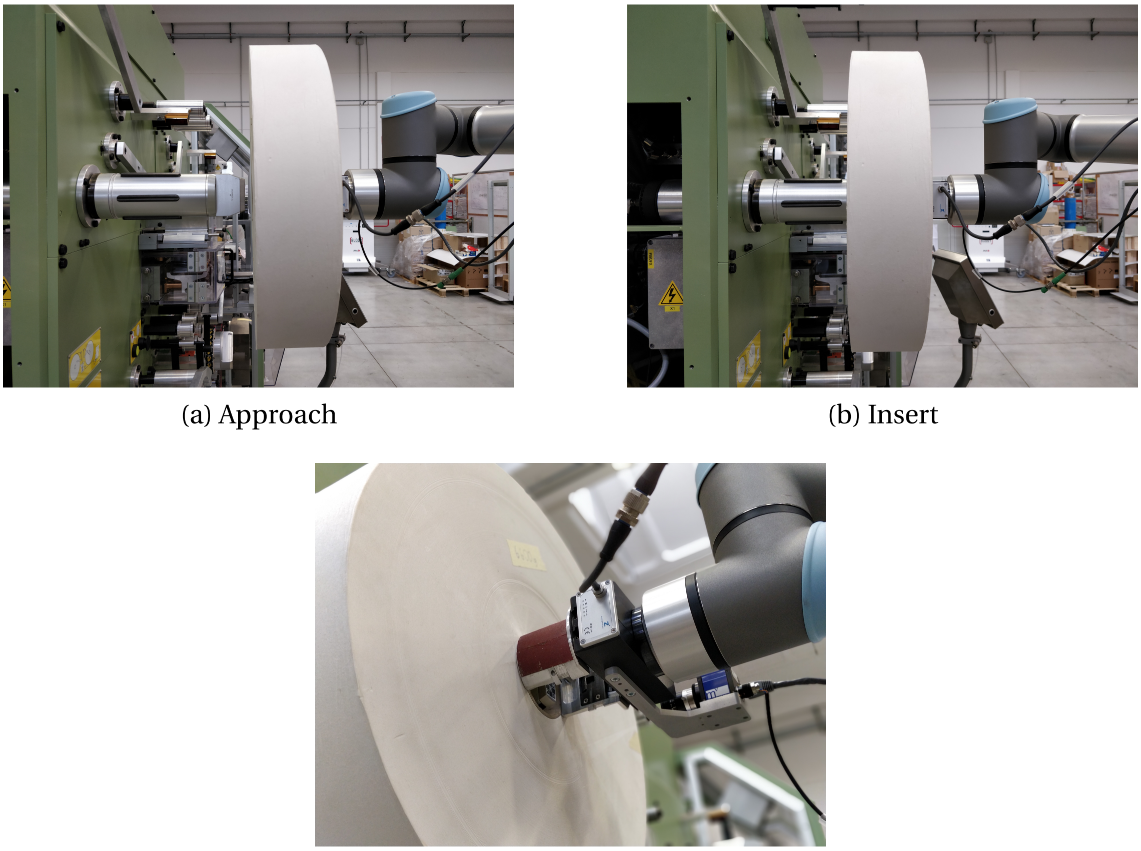
(c) Push.

(d) Grasp.

Figure 3.27: Grasping of a tag-paper-reel core.

3.5 Reel loading

The exact pose of the reel on the buffer is taken directly from the value previously saved at the end of the picking phase. In this way, there is no need to run the reel-core detection pipeline a second time. The gripper approaches the reel and inserts the fingers through a linear motion until contact. By the usual inside grasping, the reel is engaged and subsequently lifted. By recalling the marker pose, the planner computes a trajectory that brings the reel in front of the mandrel, with the reel axis aligned with the mandrel one (Fig. 3.28a). Once again, during reel manipulation, the anti-slipping constraint is active, as described in Sec. 3.3.3. As soon as the reel is inserted (Fig. 3.28b), the arm retracts enough to extract the fingers from the reel core, opens them and pushes the reel further back on the mandrel (Fig. 3.28c). Then arm #2 retracts again before moving to idle pose, followed by arm #1.



(a) Approach

(b) Insert

(c) Push

Figure 3.28: Loading of a filter-paper reel.

3.6 Fall-back procedures

By definition, a *fall-back procedure* is an operation that reverts a failed change of state and brings the system back to the original one. In an industrial context, it is important not to let any behaviour unregulated so that even errors and faults are managed in a predictable way. This enhances safety and allows the user or the maintainer to recover from these anomalous states.

In the ROSSINI use case, three types of conditions can result from a fall-back procedure:

1. *failed mission*: the assigned task could not be completed, but the AMR completely recovered and moved back to the charging station, ready for new assignments; at the same time, the work cell was not subject to significant changes that require the intervention of an external human agent;

2. *compromised mission*: similar to a failed mission but, in this case, the work cell requires the intervention of a human operator to restore the nominal state; for instance, this may be the case of a failure at the reel-loading time, since the reel may be still on board the AMR, which means that the robot is not able to pick up a new one;

3. *fatal error*: this is the worst scenario, typically linked to the undefined behaviour of the cobot during motion, which triggered a safety arrest; this may occur, for instance, when an unexpected contact is detected, or the manipulator perceives a discrepancy between the target and the current configuration due to resisting forces; in this case, an expert human operator is required to intervene to resolve the situation and resume the process, sometimes by a full restart of the application.

A schematic view of the decision flow that defines the final aforementioned condition is depicted in Fig. 3.29. The outcome is then forwarded to the task manager as feedback about the execution of the latest robot-assigned task, and a suitable action will follow according to the severity of the condition. In general, a "failed mission" simply triggers a re-assignment of the failed task, whereas the other two conditions will entail the request for missions with ID=8 in the "compromised mission" case and with ID=9 in the "fatal error" case.

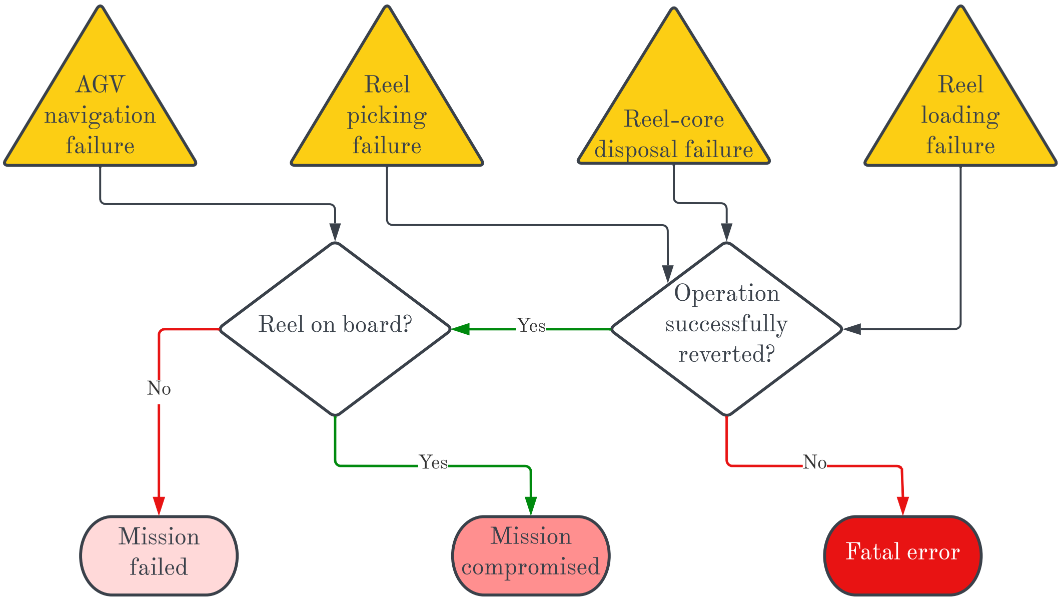


Figure 3.29: Decision tree for the outcome of a fall-back procedure.

3.7 Results and discussion

Given the technical complexity of the ROSSINI framework, a lot of effort and resources were allocated to meet the time constraints and requirements imposed by the EU commission. As a result, data from an extensive experimental campaign with statistical purposes are unavailable. Nevertheless, qualitative considerations can be made from the hundreds of experiments and tests conducted during the development of the application.

The average nominal duration of a paper-reel-change mission, from the reception of the request to the return to the charging station, is about 6 minutes. This is the time not taking into account interruptions due to the proximity of a human operator, which temporarily slows down or halts the system (e.g. SSM and SMS, respectively). The variance of the nominal time is related to several environmental factors that affect the AGV dynamic path planning and the cobots' trajectory generation. About one-third of the time (i.e. nearly 2 minutes) is spent in AGV transfer from one location to another, and one-third for the reel picking. The remaining one-third is usually split in half between the reel-core disposal and the reel-loading phases.

In general, the filter-paper-reel change yields a higher success rate with respect to the tag-paper case. From a rough analysis of the failures and successes of these missions, we can estimate the success rate to be around 90% for the former case and nearly 20% for the latter one. This is mainly due to the different rigidity of the reels. In fact, the tag-reel core tends to be more elastic, and when it is grasped from the inside, its shape often deforms from a circular to an elliptical cylinder. This makes it impossible to insert it over the mandrel, which has a circular section and very little clearance (around 1 mm) with respect to the inner diameter of the reel core. As a matter of fact, most of the failures encountered in the tag-paper-reel mission are due to this issue and, therefore, occur at the reel-loading time, specifically, at the insertion of the tag-paper reel on the relative mandrel. By disregarding these cases from the overall set of conducted experiments, that is, by employing a more rigid tag-paper reel, the success rate for the second case also settles around 90%.

The main critical issues of the mission, which entails most of the remaining experienced failures, are related to the manipulation of the reels and the trajectory planning of the cobots. The *ROS controller*, in fact, does not account for all physical limitations of the UR and for the inertia of the different payloads. An example is given by an internal feature of the UR low-level controller related to *clamping hazard*, which blocks the robot when the wrist is rotated in a configuration where a human hand could potentially be clamped between the terminal link and the forearm of the cobot. This condition is, unfortunately, very common when performing a linear motion during the loading phase when inserting the reel over the mandrel. To partially counteract this inconvenience, constraints on wrist angles at the target pose fed to the planner were manually included at this specific step of the task. This workaround is, however, too specific to the application at hand and should be replaced by a more general solution. Another example is given by the fact that the weight and inertia of an attached object during manipulation are not taken into consideration by a trajectory planner implemented with the MoveIt library. This missing feature in the API of the library sometimes leads to a SMS because the acceleration/deceleration of a heavy object (e.g. a reel) on the end-effector may be perceived as a collision due to high reaction wrenches measured by the force/torque sensor.

Another issue encountered during reel manipulation is due to the slipping of the reel when the TCP is pointing too far down. In this configuration, the friction that the gripper fingers exercise on the inner surface of the reel core is insufficient to hold it firmly. A trivial solution would be to include orientation constraints of the TCP during the whole motion. Still, the ROS MoveIt library does not provide a suitable feature to enforce such a constraint. The proposed workaround consists in exploiting the stochastic nature of the planner. Each waypoint in the trajectory computed by the planner is checked to verify that the z-axis of the TCP in that pose has a zenith angle with respect to the horizon above -15°. This axis corresponds to the longitudinal axis of the reel when it is grabbed, and the threshold value was heuristically found to prevent slippage without restricting the range of z-orientation too much. If the condition is not satisfied for all points, the trajectory is recomputed, and the safety check is repeated until the condition is met up to a certain number of attempts, after which the planning is considered failed.

In general, the causes of the failures are all related to known technical issues, which, therefore, only require time and resources to be accounted for.

As far as the fall-back procedures are concerned, the following rough observations can be made based on the heuristics emerging from the tests:

- among all failures that triggered a fall-back procedure, 10% were failed missions, 20% were compromised missions, and 70% led to a fatal error;
- 20% of failed missions were due to AGV navigation failures when approaching the wagon; these were mainly caused by the false detection of obstacles along the path due to light reflection on the ground, which confuses the vision-based navigation of the MiR500 AGV;
- 70% of failed missions were due to eye-on-hand camera technical issues related to hardware and communication, which eventually led to a missed detection of targets (either the wagon marker or the reel core);
- 10% of failed missions were due to issues at the reel-depositing stage during reel picking, which, however, was possible to successfully revert by repositioning the reel in its original pile on the wagon;
- 70% of compromised missions were due to the failure at the extraction stage during filter-paper-reel-core removal; this is caused by a slight misalignment between the actual mandrel axis and the one obtained by the relative marker pose estimation; if friction overcomes a user-defined threshold, the operation stops and the robot remains operative, but it inhibits the next step, i.e. the reel loading;
- 30% of compromised missions were due to miscellaneous issues at the reel-loading stage, such as camera failure in the detection of visual markers or unexpected contacts during the insertion of the new reel over the mandrel; the latter is also caused by the inaccuracy in the estimation of the marker poses that determines the exact location of the mandrels;
- 50% of fatal errors were due to the aforementioned UR limitations not considered by the planner, such as the clamping hazard; these kinds of events trigger a

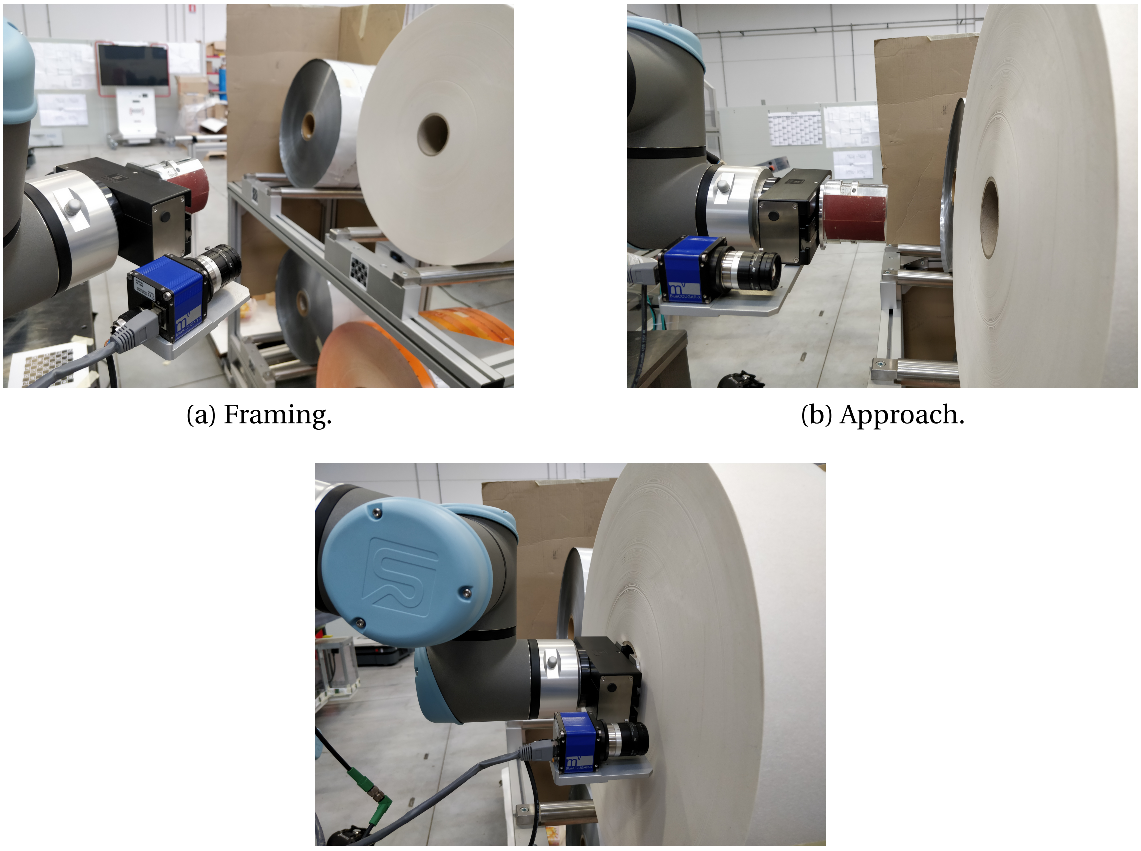
protection stop, which can only be manually disabled by acting on the cobot controller onboard and requires the operator to move the cobot outside the critical configuration manually;

- 50% of fatal errors were due to issues during reel manipulation, such as unexpected collisions (very rare) due to modelling and pose estimation errors or irreversible problems during the deposit of the reel on the buffer during reel picking or the insertion of the reel over the mandrel during reel loading; the latter case occurred more often, especially with the tag-paper reel due to the aforementioned rigidity issues.

The following Sec. 3.7.1 gives a more thorough analysis of the vision-based reel-core detection phase [48].

3.7.1 Reel-core detection test analysis

In order to test the efficacy of the proposed methods for reel-core detection, we assigned the AMR the task of picking each reel many consecutive times. Specifically, the operation consists in approaching the wagon, scanning the targeted pile from a predefined perspective and, if found, inserting the gripper in its core enabling grasping from inside, as shown in Fig. 3.30.



(a) Framing.

(b) Approach.

(c) Insertion.

Figure 3.30: Picking preliminary operations.

The pose estimation is considered successful if no issues are detected during the insertion phase, as described in Sec. 3.3.3.

The wagon approach operation introduces a small uncertainty due to the positioning error of the AGV with respect to the stationary wagon. This inconvenience is exploited to enhance the robustness of the process against small changes in the resulting framing perspective, which is always relative to the cobot base frame. Moreover, to avoid being influenced by the position of the reel on the pile, the tests are carried out over reels at different locations on the shelf, from the one at the front to the one at the back.

Table 3.3 shows the results of the whole experimental campaign in terms of computational time and final outcome.

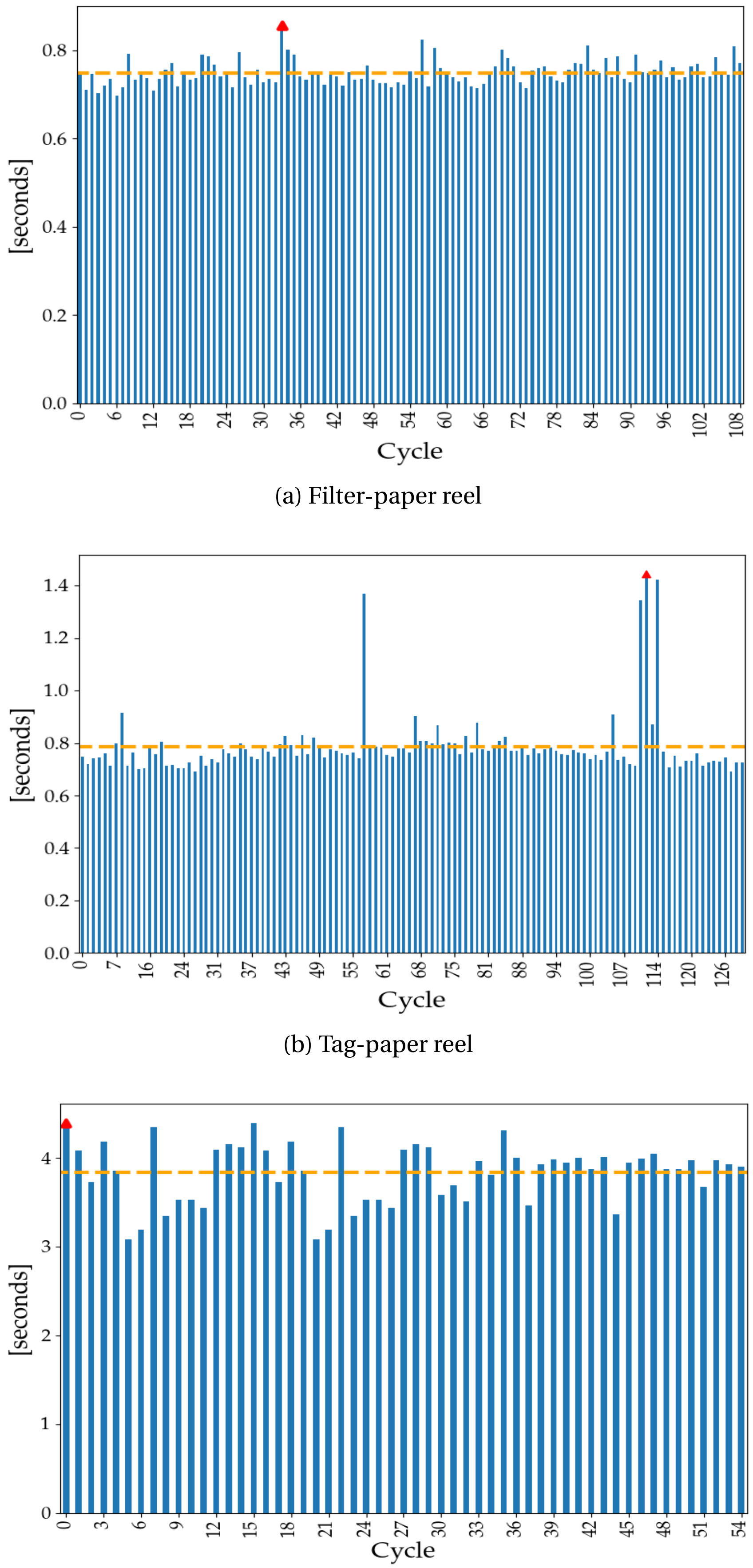
The contour method was successfully employed in all cycles of paper-reel and tag-reel detection, except for a single iteration in the latter, where none of the proposed approaches led to success. For the outer-envelope reel, instead, the Hough method proved to be the most suitable given the zero failures achieved. As expected, the contour method is the fastest, allowing the system to provide a pose estimate in less than 1 second on average, as shown in Fig. 3.31. The largest values are related to an adverse light condition that pushed the algorithm to auto-tune the parameters before computing a valid response, as described in Sec. 3.3.2.1. The Hough method, despite being very effective and robust, is also the slowest by a large amount. The iterative procedure that looks over all possible combinations in the given (reduced) search space introduces a computational burden which is hard to avoid. However, the outer-envelope reel is the one which takes longer to be consumed by the automatic machine. Therefore, a little extra time is still acceptable as it does not significantly impact the overall reel-changing time. Moreover, this type of reel was not considered for manipulation in the ROSSINI use case. For this reason, the effort to improve performance, in this case, was limited to a minimum.

Table 3.3: Experiments results.

Reel Type	**Total Trials**	**Average Time**	**Maximum Time**	**Failures**
Paper	109	0.749 sec	0.857 sec	0
Tags	118	0.786 sec	1.445 sec	1
Outer Envelope	55	3.831 sec	4.392 sec	0

3.7.1.1 Tuning and user-defined settings

The excellent outcomes achieved were the result of a fine-tuning session that allowed us to be particularly robust in the current scenario. Despite the auto-tuning of a few parameters to cope with the varying light condition, there is a set of fixed parameters which must be suitably defined by the user. They do not require any special skill or expertise and, thus, can be easily managed by a non-expert operator. However, they do need a short trial-and-error session as they are strictly related to the expected kinds and variability of targeted reels and how they appear in an image. These settings are mainly used to discard potential false positives and restrict the search space over valid outcomes, but also to speed up computational time (especially the ones used by the Rbrown and Hough method). In accordance with this concept, they express ranges of values that are considered valid depending on a rough qualitative analysis carried out by the user beforehand. In Tables 3.4-3.6, the settings used for this experimental campaign are reported.



(a) Filter-paper reel

(b) Tag-paper reel

(c) Outer-envelope reel

Figure 3.31: Computational times for reel-core-pose estimation. The dashed line represents the average time, and the red marker is the maximum time experienced (see Table 3.3).

Table 3.4: Experiments settings for the filter-paper reel. **R**: RBrown, **H**: Hough.

Name	**Value**	**Used by**[9]	**Description**
min_major_axis	400	R,H	Minimum major-axis length in pixels of the ellipse resembling the reel core.
max_major_axis	1100	R,H	Maximum major-axis length in pixels of the ellipse resembling the reel core.
min_aspect_ratio	0.7	R,H	Minimum aspect ratio of the ellipse resembling the reel core.
max_aspect_ratio	1.0	R,H	Maximum aspect ratio of the ellipse resembling the reel core.

Table 3.5: Experiments settings for tag-paper reel. **C**: contour, **R**: RBrown, **H**: Hough.

Name	**Value**	**Used by**	**Description**
min_contour_area	3e+5	C	Minimum area in pixels^2 circumscribed by the contour of the ellipse resembling the reel core.
max_contour_area	3e+6	C	Maximum area in pixels^2 circumscribed by the contour of the ellipse resembling the reel core.
max_goodness	15	C	Maximum value of "goodness" for a valid ellipse. See Sec. 3.3.2.1.
min_major_axis	1000	R,H	See Table 3.4.
max_major_axis	1400	R,H	See Table 3.4.
min_aspect_ratio	0.7	R,H	See Table 3.4.
max_aspect_ratio	1.0	R,H	See Table 3.4.
rotation	70	R,H	Rotation of the ellipse (e.g. its main axis) with respect to the horizontal axis in degrees.
rotation_span	15	R,H	Tolerance for *rotation* in degrees.

Table 3.6: Experiments settings for the outer-envelope reel. **R**: RBrown, **H**: Hough.

Name	**Value**	**Used by**	**Description**
min_major_axis	1000	H,R	See Table 3.4.
max_major_axis	1400	H,R	See Table 3.4.
min_aspect_ratio	0.7	H,R	See Table 3.4.
max_aspect_ratio	1.0	H,R	See Table 3.4.
rotation	70	H,R	See Table 3.5.
rotation_span	15	H,R	See Table 3.5.
min_edge_size	100	H,R	Minimum candidate edge length in pixels.
sigma0	4.0	H,R	Initial blurring value for smoothing the image before edge detection.
min_feat	4	H,R	Minimum number of detected edges.

Chapter 4

SENECA use case: bin scanning and cleanliness classification

In this chapter, we describe the procedure and algorithms involved in the SENECA use case starting from bin identification in Sec. 4.1, moving to bin surface scanning in Sec. 4.2 and concluding with Sec. 4.3, dedicated to bin cleanliness classification.

To the best of the author's knowledge, the automatization of the surface scanning of a pharmaceutical bin is an original industrial implementation of known generic techniques that employ a given 3D model of the scanned object. On top of the implementation and integration of this procedure over the ROSSINI platform used to perform the experiments, our work extends the algorithmic design to consider the presence of a robotic manipulator that introduces constraints to the feasible pose of the camera used for scanning. The second original contribution in this context can be found in the generation of a binary classifier model using a popular Convolutional Neural Network to distinguish clean surfaces from dirty ones purely based on data (e.g. images).

Both contributions are to be included in a work-in-progress article [55] not yet published.

4.1 Bin identification

The main requirement for the bin-identification procedure was to be independent of the type of bin that will be loaded inside the washing cabin in the future. Despite the fact that the experimental campaign was carried out on a single pharmaceutical bin, the classification of its topology remains an interesting topic that deserves to be analysed in view of future developments.

4.1.1 Methodology

Two methods were evaluated to carry out the bin detection and identification: pose estimation from a 2D image and point cloud registration. The common goal was to define the exact shape of the bin at hand quickly and its relative pose with respect to the inspection/cleaning robot base-frame through a vision system.

4.1.1.1 Method 1: point cloud registration

Point cloud registration is the name used in literature to address the process of matching a polygon mesh[1] obtained from a 3D CAD model of an object with a measured point cloud representing the same object as seen from a 3D sensor such as a stereo camera or a laser scanner. The topic has been widely studied and discussed due to its practical implications, and notable results have been achieved [56],[57],[58]. However, this approach may not yield sufficient repeatability due to the noise in the measurements. It may produce positioning errors up to a few centimetres depending on the resolution of the employed sensor and the complexity of the observed scene and target. Moreover, depth sensors needed to produce this kind of data are typically more expensive than 2D cameras, especially when a large resolution is required. On the other hand, this approach's main advantage was that no additional mechanical intervention or application of a visual tag on the bin was required.

4.1.1.2 Method 2: pose estimation from a 2D image

Pose estimation is generally the task of detecting the 6D pose of an object, which includes its location and orientation, from a single 2D (usually coloured) image. Because a traditional 2D camera is substantially a bearing sensor, given an object in a framed scene, it is not possible from a single image to extrapolate its pose using only the bare data (i.e. the pixels) without additional external information. Feature matching + homography is a common bundle of techniques that allows estimating the pose of a template object of known dimension in a cluttered scene where the object appears in an arbitrary pose (up to a certain level). The large availability of feature-point detectors and descriptors and the many variants of RANSAC optimization to solve the homography step made this approach suitable for many industrial scenarios. Nonetheless, the robustness and efficacy of the method strongly depend on the context, such as the light conditions and the structure of the environment (repeatability, noise, etc.).

An even simpler approach is the one based on the use of well-defined patterns, such as chessboards, that may be easily and distinctively recognized in an image. The *visual markers* (or *tags*) have known size and dimensions and, once detected in the image, their pose can be immediately calculated with respect to the frame of the camera that took the shot. Differently from feature-based matching, subject to the effect of noise, light and clutter of the scene, visual-tag-based pose estimation usually yields millimetric accuracy with little computational effort and very fast response. On the other side, not all use cases are suitable for the application of a visual tag on the surface of each object whose pose should be estimated.

4.1.2 Implementation

The strategy of estimating the pose through a visual marker, already successfully adopted in ROSSINI for the wagon and the automatic machine as described in Sect. 3.3-3.5, is proposed again here. On top of the advantages described in Sect. 4.1.1.2, another reason that supported this choice is linked to the availability of the whole implementation from the ROSSINI framework, which considerably accelerated the development

[1] A *polygon mesh* in 3D computer graphics and solid modelling is a collection of vertices, edges and faces that compose a polyhedral object. The faces are usually triangles (triangle mesh), quadrilaterals (quads) or other simple convex polygons (n-gons).

process. Moreover, markers with different characteristics could also be used to identify a particular topology of the bin, providing information about both the pose and the shape at the same time.

As a first step, the AGV must move close to the bin, located in a fixed position in the work cell. The approach motion is subject to uncertainty due to drift and mapping errors and can lead to a positional error of a few centimetres when there is no 3D marker that helps the platform to dock precisely to a work location. This inaccuracy is actually helpful to mimic a possible error in the placement of the bin inside the washing cabin. In fact, while the envisioned robotic arm should be fixed to the ceiling of the cabin, the bin shall be inserted by a human operator, thus subject to small positioning errors if a mechanical guide is not present.

To avoid collision and suitably include the bin in the planning scene, however, we need a precise relative pose between the bin and the inspection robot. To this purpose, a ChArUko marker is applied on one of the vertical sides of the bin. Thanks to the visual marker and using the industrial 2D camera already part of the ROSSINI platform, the tag is framed, and its precise position is measured with millimetric accuracy (< 2 mm positioning error). Knowing a priori the fixed transformation between the marker frame and the bin mesh frame makes it possible to spawn the collision object in the planning scene, as shown in Fig. 4.1.

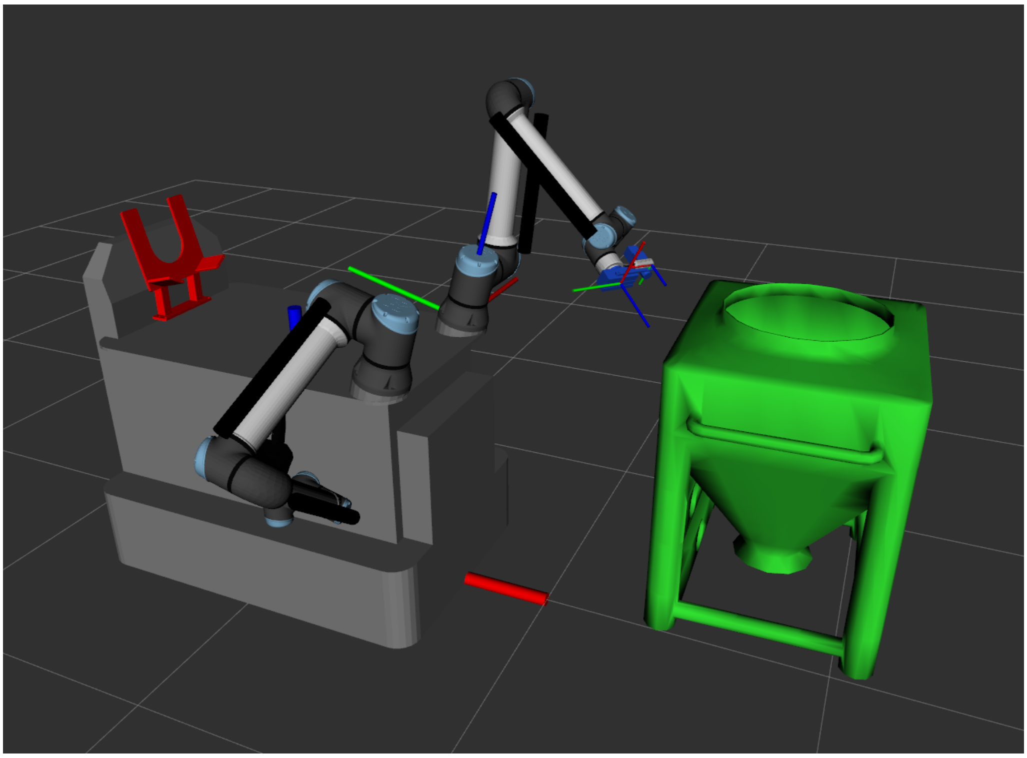


Figure 4.1: Virtual twin used for collision-free dynamic trajectory planning.

This operation is carried out as the initial step of the bin-scan mission in both offline and online modes, as discussed in detail in Sect. 4.2.2.1-4.2.2.2, respectively. In the former case, the marker pose is loaded rather than resulting from an actual scan of the marker with the camera and can be manually modified in a configuration file.

4.2 Bin scanning

4.2.1 Methodology

The algorithm used for this phase is an adaptation of the recent work from [59], which puts together the results from [60] about a non-voxel[2] approach for viewpoints generation, and from [45] for viewpoints sorting. Suggestions and insights from [61] are also used by the author to obtain satisfactory results in a realistic pipeline. In this section, we describe the main steps that compose the algorithm as well as the introduction of new kinematic constraints related to the employment of a robotic manipulator to reach the computed viewpoints.

4.2.1.1 Viewpoint generation pipeline

Given a CAD 3D model of the bin, the object is loaded as a mesh with triangular faces and converted into a uniform point cloud. The amount of points can be arbitrarily defined as a percentage of the number of faces. The bin mesh reference system is assumed to have the z-axis pointing upwards, as shown in Fig. 4.2.

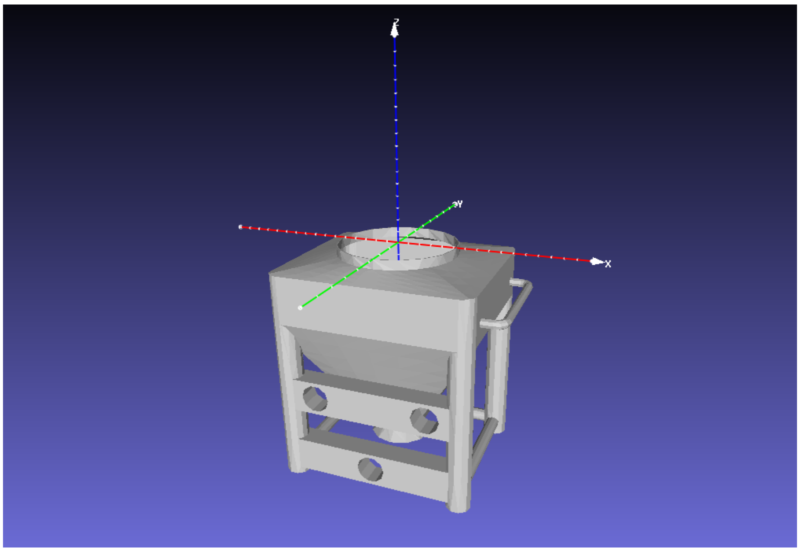


Figure 4.2: Bin mesh with the reference system.

Each so-called *seed point* $\mathcal{S} = (S_x, S_y, S_z)$ is associated with the closest triangular face of the original mesh, therefore called *seed face*, and used to compute the ViewPoint (VP) pose. Considering the pose as a 3D reference system with an origin and an orientation expressed in the form of a homogeneous transformation $\mathbf{H} \in \mathbb{R}^4$, we consider two different scenarios: internal and external scanning.

In the former one, all the poses irradiate from a fixed point located at the origin of the bin reference system, which corresponds to the centre of the upper hole $\mathcal{O} = (O_x, O_y, O_z)$ (Fig. 4.3). Then, the direction $\mathbf{q}$ of each pose, which coincides with the z-axis of the relative reference system, is oriented such that it points towards a different seed point (only the internal surface is used to generate the seed points in this case). Hence, defining the number of seed points with N, we have:

[2] In 3D computer graphics, a *voxel* is a value on a regular grid in three-dimensional space.

$$\mathbf{q}_i = \frac{(\boldsymbol{\mathcal{S}}_i - \boldsymbol{\mathcal{O}})}{\overline{(\boldsymbol{\mathcal{S}}_i - \boldsymbol{\mathcal{O}})}} \quad , \quad \text{for} \quad i = 0, \dots, N-1 \tag{4.1}$$

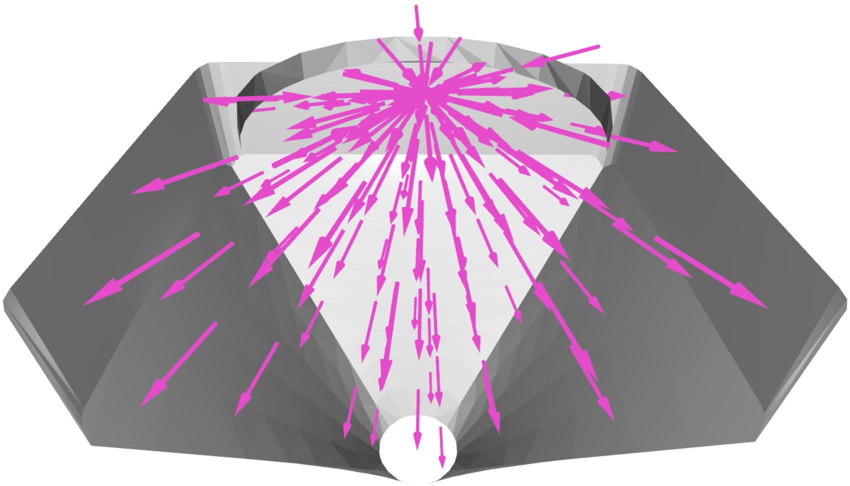


Figure 4.3: Internal viewpoints (132) irradiating from the bin mesh origin.

In the latter case, the z-axis is aligned with the corresponding seed-face normal but with the opposite direction (Fig. 4.4). Hence, by naming $\mathbf{n}_i$ the normal of the i-th seed face with origin in $\boldsymbol{\mathcal{S}}_i$:

$$\mathbf{q}_i = -\mathbf{n}_i \quad , \quad \text{for} \quad i = 0, \dots, N-1 \tag{4.2}$$

Given the direction $\mathbf{q}$ of each viewpoint in both cases, the origins of the viewpoint frames $\mathbf{t}_{i,j}$ are located at a suitable number D of constant distances $d_0, \dots, d_{D-1}$ from the seed point $\boldsymbol{\mathcal{S}}_i$:

$$\mathbf{t}_{i,j} = \boldsymbol{\mathcal{S}}_i - d_j \mathbf{q}_i \quad , \quad \text{for} \quad i = 0, \dots, N-1 \quad \text{and} \quad j = 0, \dots, D-1 \tag{4.3}$$

The distances must be selected within the range of valid depths of field of the sensor and should appear in the list in order of preference. This is important because in the end only the first reachable viewpoint along each direction q_i will be kept. According to the application, it may be convenient to prioritize closer or farther perspectives but at the same time, those positions may entail a collision or generally be unfeasible. For this reason, more options, i.e. $D > 1$, increase the chance of using each direction q_i.

The remaining axes are calculated systematically to keep the perpendicularity with the z-axis, i.e. $\mathbf{q}$, according to the method proposed by Kundu *et al* [62]. Let's initially assume that the viewpoint's coordinate frame is aligned with the mesh coordinate frame. The objective now becomes finding the rotation matrix $\mathbf{R}$ that aligns the initial z axis, denoted as $\mathbf{p}$, to the target z axis, or direction of the viewpoint, i.e. $\mathbf{q}$, and it is obtained by Eq. 4.4:

$$\mathbf{r} = \mathbf{p} \times \mathbf{q} \tag{4.4}$$

$$\mathbf{R} = \mathbf{I} + \lfloor \mathbf{r} \rfloor + \frac{\lfloor \mathbf{r} \rfloor^2}{1 + \mathbf{p} \cdot \mathbf{q}} \tag{4.5}$$

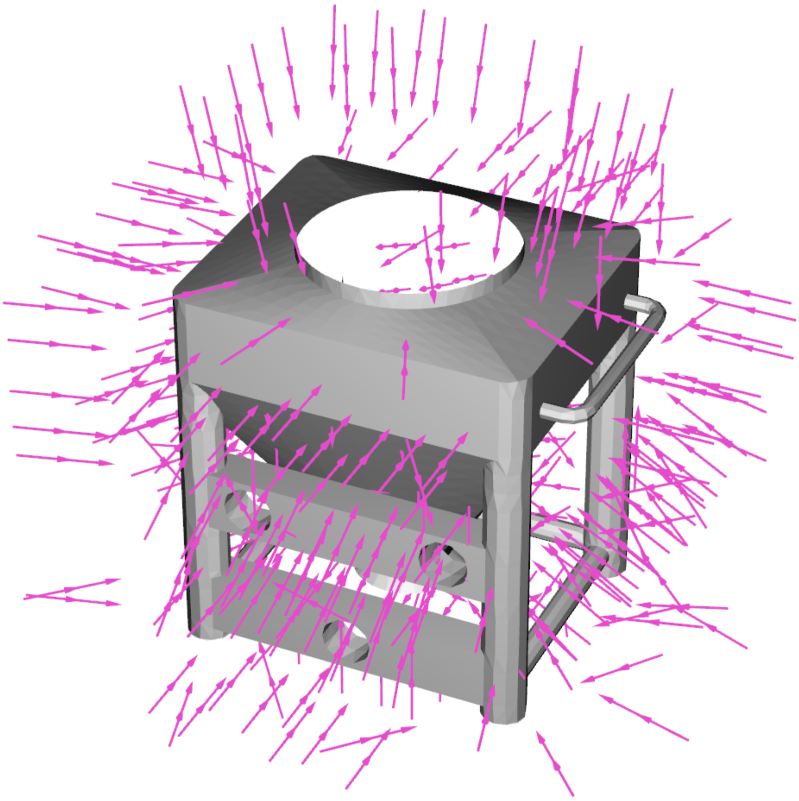


Figure 4.4: External viewpoints (406) aligned with seed-faces normals.

being $\lfloor\cdot\rfloor$ the skew-symmetric matrix operator. The resulting homogeneous transformation matrix then becomes:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.6}$$

An example of two viewpoints resulting from this process is shown in Fig. 4.5.

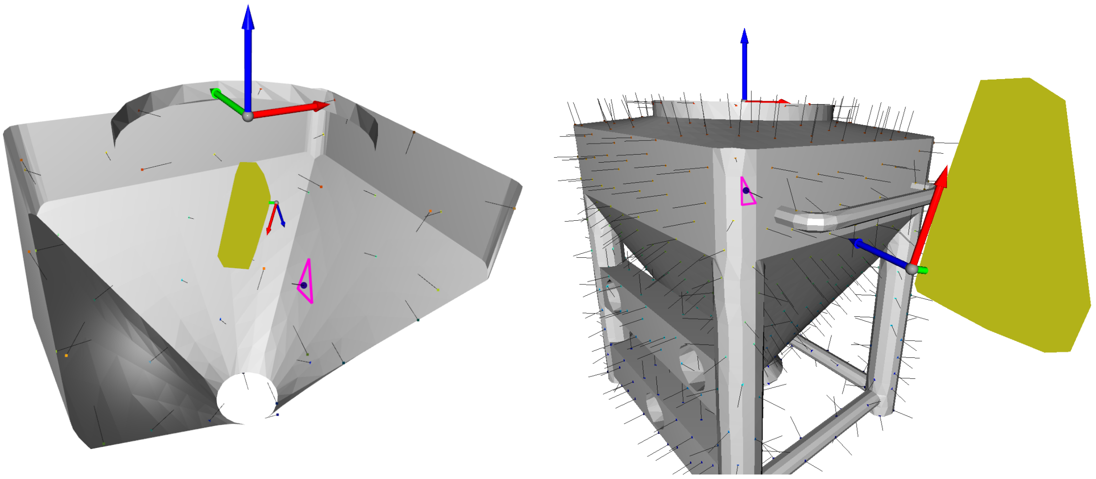
Each viewpoint can then be kept or discarded according to any combination of the following conditions, listed in order of increasing computational effort:

1. **height check**: it excludes all those viewpoints whose origin is below a certain threshold, resembling the impossibility of placing the camera underground or below a certain height;

2. **distance check**: it excludes all viewpoints whose origin is farther than a given distance from a convenient 3D point expressed in the coordinate frame of the bin; this helps, for instance, to discard all those points beyond the reach of a manipulator by placing the reference point at the origin of the robot base frame;

3. **occlusion check**: it excludes a viewpoint if a ray, starting at the viewpoint origin and ending at the centroid on the corresponding viewpoint seed face, intersects any other triangle belonging to the mesh.

4. **collision check**: after loading the mesh of the camera (or the assembly of the tools) and positioning it according to each viewpoint pose, the viewpoint is discarded if a collision between the camera mesh and the bin mesh is detected.



(a) Internal surface.

(b) External surface.

Figure 4.5: Bin mesh with normals and a viewpoint example. Coloured dots are the seed points obtained from the point cloud. The dark blue dot is the seed point used for the current viewpoint. The magenta triangle around the seed point is the corresponding seed face. The yellow block is the simplified encumbrance of the eye-on-hand tools assembly.

Once this initial filtering is complete, it is possible to resort to a simulation environment to test the actual feasibility of the remaining viewpoints. This is possible by spawning the bin and the robot in a realistic relative position and using the dynamic planner described in Sect. 2.3 to test each viewpoint pose (Fig. 4.1). The procedure may be long, but it is a readily available solution to exclude unreachable poses considering both overall dimensions and the robot's kinematic constraints. It is based on simple evidence that if a pose is reachable, the planner will find a solution to the planning problem. If not, that pose is discarded.

The last step of this phase consists in computing the visible faces from each remaining viewpoint. This is done by collecting all the visible faces inside the camera's frustum using ray casting techniques that include the camera's intrinsic parameters to define the cone of view according to the camera pin-hole model. A face is considered properly visible if its distance from the camera origin is within a predefined range expressing the depth of field of the sensor, and the glance angle is above a threshold heuristically defined according to the characteristic of the sensor and of the inspected surface (see an example in Fig. 4.6). This helps ignore faces that are too far or too close and, therefore, probably out of focus or too tilted to be properly observed.

An example of the application of this procedure using all the filters is shown in Fig. 4.7. In particular, it may be observed how the kinematic constraints and the encumbrance of both the arm and the tool together play a major role in excluding the viewpoints facing one side of the internal surface. Likewise, the opposite side of the one approached by the robot cannot be reached by the kinematic chain, and thus, no viewpoints appear there. In general, for the same point-cloud-sampling resolution, more viewpoints are left in the external surface case due to the larger number of faces and the higher freedom of movement that the robot has compared to the narrow space inside the bin.

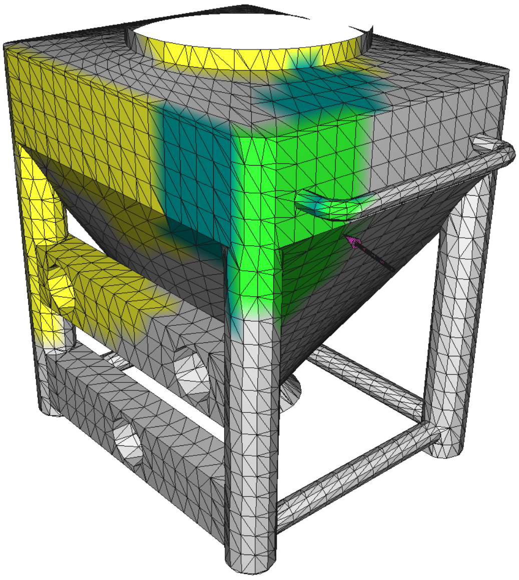
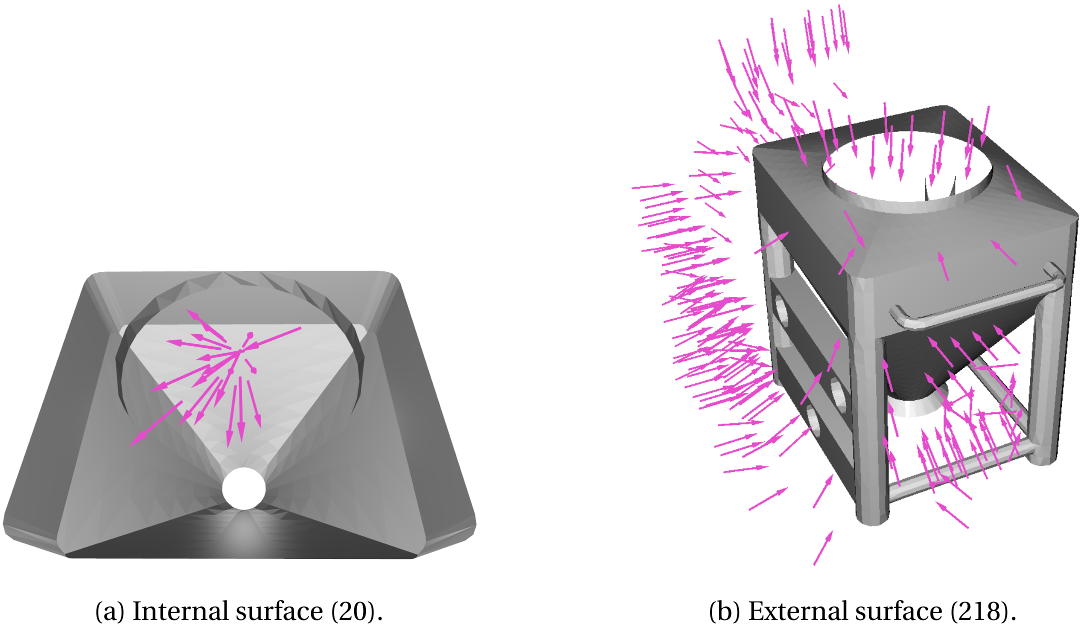


Figure 4.6: Bin mesh with visible faces from a viewpoint. Green triangles are the valid visible faces. Cyan triangles are faces with an invalid glance angle. Yellow triangles are the faces that are outside the visible range. Grey triangles are the faces which are not visible from the viewpoint, represented by the magenta arrow indicating its direction (i.e. z-axis), and whose base coincides with the viewpoint origin.



(a) Internal surface (20).

(b) External surface (218).

Figure 4.7: Resulting viewpoints after filtering.

4.2.1.2 Viewpoints selection

The viewpoints selection is based on two optimization algorithms called *Greedy Area (GA)* and *Simulated Annealing (SA)*, respectively, better discussed in Sec. 3.2.2 of [59].

Both methods rely on the existence of a *measurability matrix*, $\mathbf{C} \in \mathrm{R}^{m \times n}$, a two-dimensional binary array where the m rows correspond to the mesh faces and the n

columns to the viewpoints from the original set, i.e. the set of viewpoints that are left after the filtering and the reachability check. Each entry $c_{i,j}$ is equal to 1 if the i-th face is properly visible from viewpoint j; otherwise, it has a value of 0. This matrix can also be used to calculate the total visible area from all viewpoints and the visible area from every viewpoint.

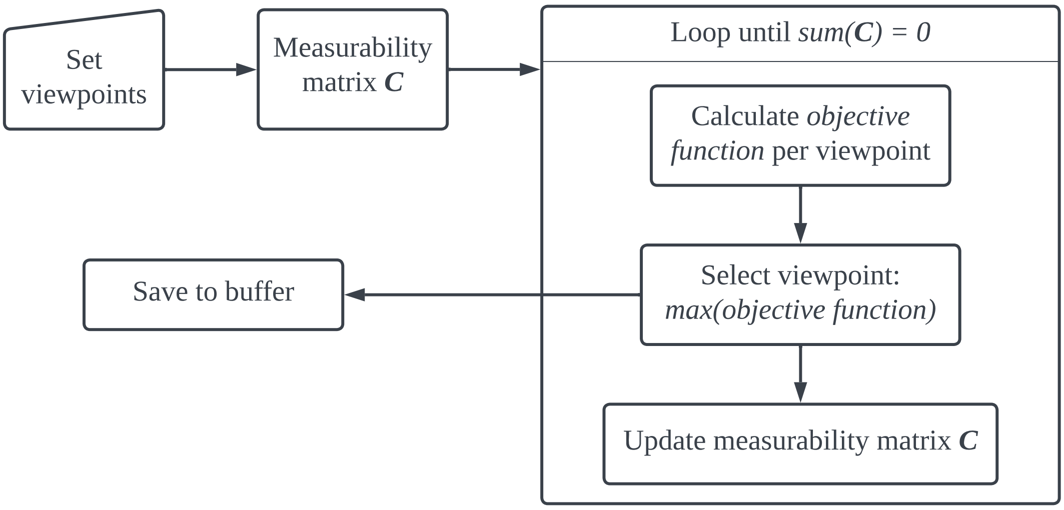

Set viewpoints
Measurability matrix ***C***
Loop until *sum(**C**) = 0*
Calculate *objective function* per viewpoint
Select viewpoint: *max(objective function)*
Update measurability matrix ***C***
Save to buffer

Figure 4.8: Flowchart of the greedy area method.

In the GA method, the viewpoint that maximizes the objective function is selected at each iteration. The objective function, in this case, is the sum of the area of the visible faces from a single viewpoint. Therefore the selected viewpoint is the one from which it is possible to measure the largest surface, hence its name (Greedy Area). After the selection, the rows corresponding to the faces that were already measured by the latest best viewpoint are zeroed so that those faces will not play any more roles in the next iteration. The same routine continues until the measurability matrix is filled with only zeros. Consequently, this optimization process outputs the minimum number of viewpoints necessary to cover the entire surface that would be visible using the whole original set of viewpoints. A schematic view of the procedure is shown in Fig. 4.8, whereas an example of the application of this optimization is illustrated in Fig. 4.9.

Instead, the SA method, starting from the subset obtained by the GA method, aims to reduce it further by a desired percentage. This is done by iteratively and randomly swapping viewpoints from the current subset (initialized with a portion of the GA subset) to the original set of all available viewpoints and tracking the combination of viewpoints that maximizes the visible area, which represents the solution of this optimization problem. Let A be the set of all the n viewpoints left after the filtering discussed in Sect. 4.2.1.1, and B a subset of A. B is initialized as a desired percentage p_{SA} of randomly picked elements in the subset C of A obtained with the GA method, with dimension n_{GA}, namely:

$$\begin{aligned} & n_{SA} < n_{GA} \leq n \\ & n_{SA} = \frac{p_{SA}}{100} n_{GA} \quad , \quad \text{with} \quad 0 < p_{SA} < 100 \\ & B_0 = \left\{ \mathbf{H}_0, \ldots, \mathbf{H}_{n_{SA}} \right\} \in_R C \subset A \end{aligned} \tag{4.7}$$

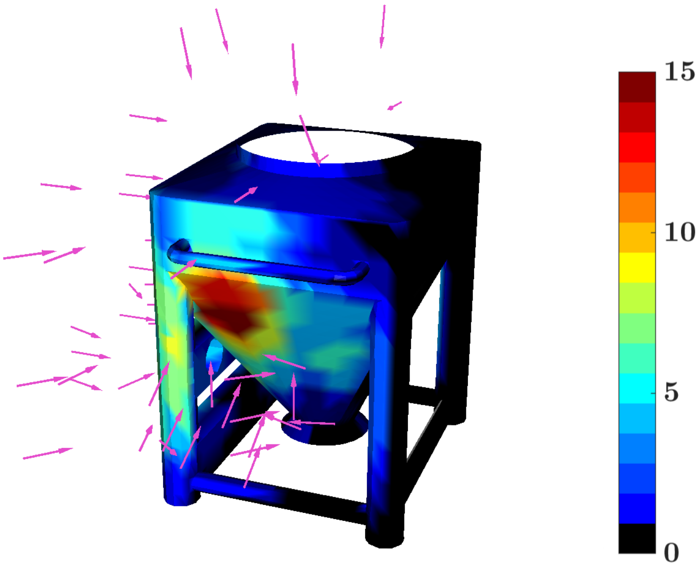


Figure 4.9: Viewpoints selected by the GA method (61). The colour map describes how many times a face was seen from a different viewpoint. The maximum value depends on the mesh and the settings.

At each iteration k, n_{swap} viewpoints are swapped between the set B^*, which consists in the current best combination of viewpoints found (i.e. best solution), and the set of unused viewpoints $\Delta = A \setminus B^*$. Then, the new solution contained in B_k is evaluated through a cost function which calculates the missing area shown in Eq. (4.8):

$$Cost(B_k) = 1 - \frac{Area(B_k)}{Area(A)} \tag{4.8}$$

where $Area(\cdot)$ is a function that computes the total amount of visible area using the viewpoints in the given set and is bounded between 0 and 1 since $B_k \subset A$. Given the new cost, if the new set of viewpoints contained in B_k provides a larger visible surface (i.e. lower cost), it replaces the current one, i.e. $B_k \mapsto B^*$. On the contrary, that is if the cost is higher, B_k is either kept or discarded according to the *MAC* [63]. Following this technique, we first calculate the MAC value according to Eq. (4.9):

$$\begin{gathered} \delta = Cost(B_k) - Cost(B^*) > 0 \\ 0 < MAC = e^{\frac{\delta}{T_k}} < 1 \end{gathered} \tag{4.9}$$

being T_k the *temperature* parameter at iteration k, usually heuristically chosen or resulting from an optimal tuning session. The MAC value is then compared to a real number randomly sampled from a uniform distribution from 0 to 1, and the new solution B_k replaces the current one B^* if the random number is lower than the MAC value.

Both the values of n_{swap} and T follow an exponential decay $\gamma \in \mathbb{R}^+$ as in Eq. (4.10) to allow a higher exploration at the beginning.

$$\begin{gathered} n_{swap,k} = n_{swap,0} * e^{-\gamma * k} \\ T_k = T_0 * e^{-\gamma * k} \end{gathered} \tag{4.10}$$

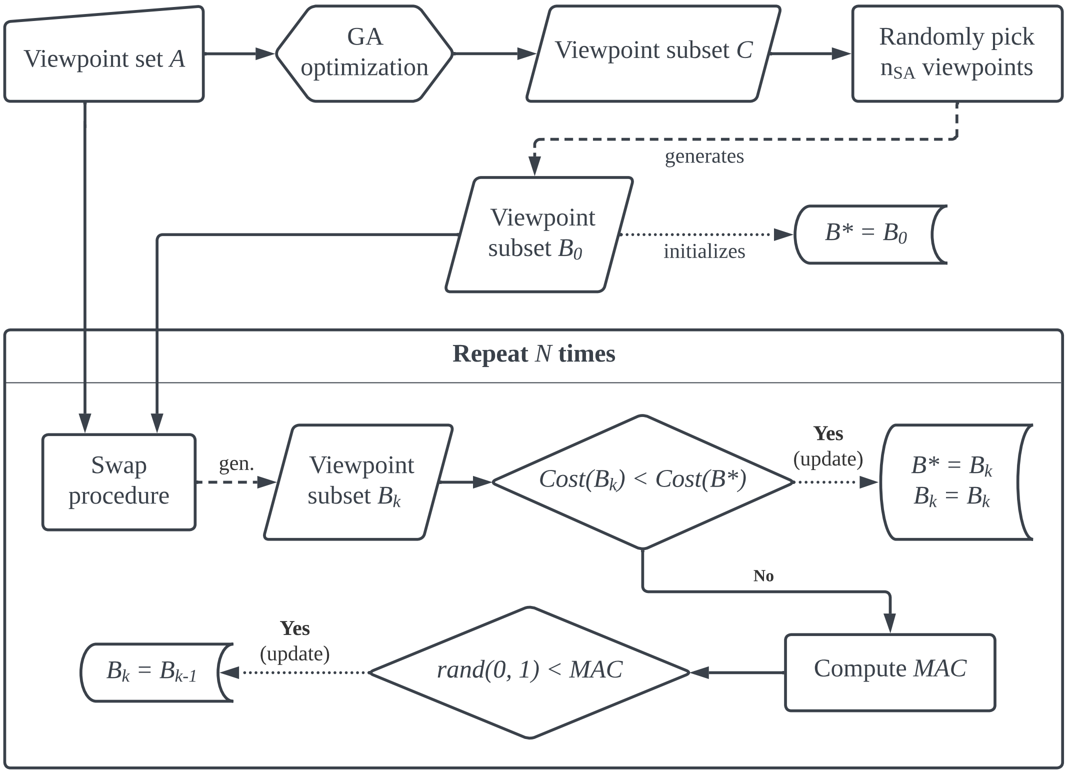


Figure 4.10: Flowchart of the simulated annealing method.

This procedure decreases the number of viewpoints at the expense of a very little loss of visible area (ideally 0%), and its efficacy strongly depends on the tuning parameter of both the MAC number and exponential decays involved in its implementation. The stopping criterion solely depends on the number of iterations defined by the user unless a zero cost is obtained at any step. Similarly, a schematic view of the procedure is shown in Fig. 4.10, whereas an example of the application of this optimization is illustrated in Fig. 4.11.

4.2.1.3 Viewpoints sorting

Given the resulting subset of viewpoints, the last step consists of finding the best order in which they should be visited to minimize the overall path length. This problem is known in the literature as the *Traveling Salesman Problem*, and there are several solutions to it [64], [65]. The chosen one is the same as proposed by [59], that is, the *Ant Colony Optimization (ACO)* [45], inspired by this insect behaviour.

This optimization only requires a *distance matrix* as input which registers the cost of moving from viewpoint i (row-wise) to point j (column-wise). To populate this matrix the procedure is similar to the reachability test carried out resorting to the simulation environment. In fact, all combinations i-to-j and vice-versa are tested, and the length of the planned path is used as the cost and, thus, the entry value.

Given the distance matrix, the outcome of the ACO is the order of filtered (Sect. 4.2.1.1) and selected (Sect. 4.2.1.2) viewpoints that provides the shortest path passing through all of them only once.

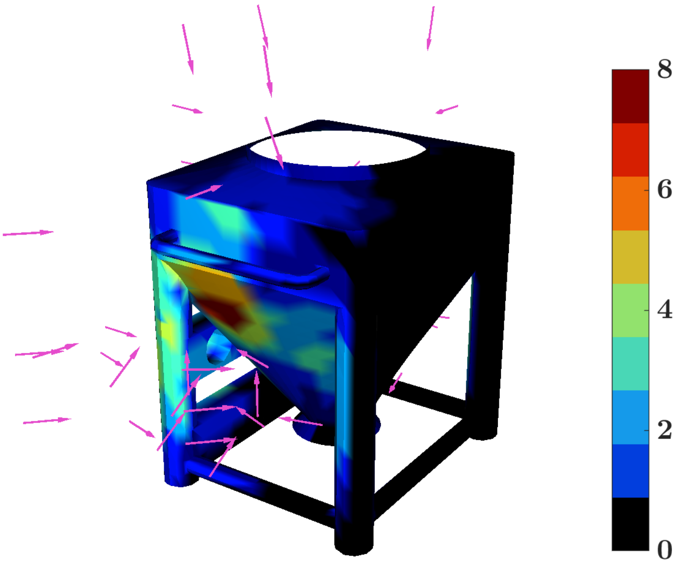


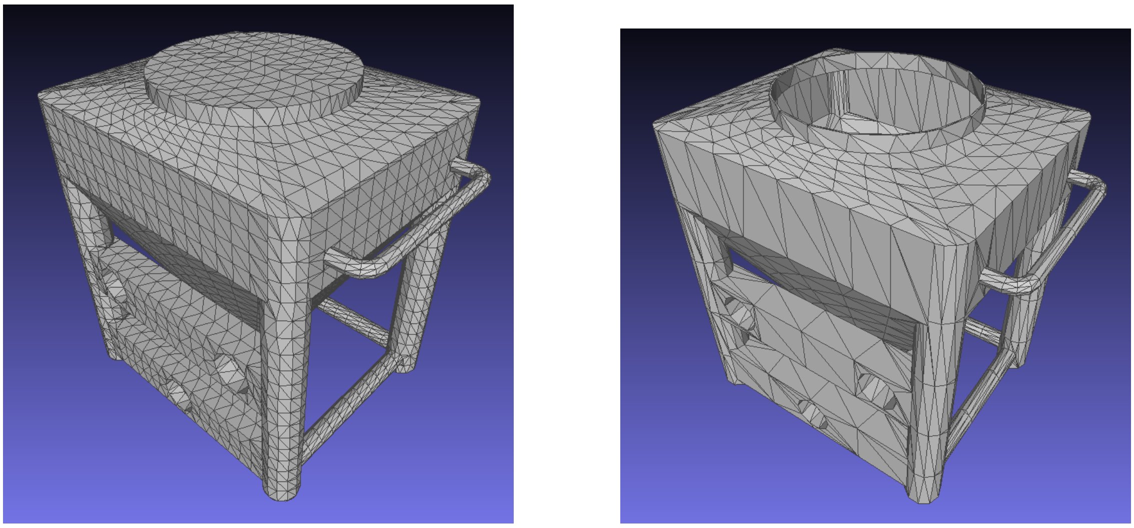
Figure 4.11: Viewpoints selected by SA method with 15% reduction (51). The colour map describes how many times a face was seen from a different viewpoint. The maximum value depends on the mesh and the settings.

4.2.2 Implementation and results

For a particular bin, the view poses that can be generated and optimized can, in principle, be computed once and for all. However, the integration of robot kinematics and obstacle avoidance in the reachability of those configurations requires the knowledge of the relative pose between the robot and the bin. This pose was subject to changes as discussed in Sec. 4.1.2. As a consequence, a hybrid method was considered. When the robot finds the bin for the first time, it aborts the operation and requests the offline generation of the view poses and optimal path considering the bin in that specific location. The offline procedure may last several minutes (or even more than 1h) according to the user settings, and it saves a model that can then be loaded and used online. At the next iteration, the process checks for the presence of this model, and if found, it compares the current measured relative pose of the bin with the one used in the offline simulation. Up to a certain tolerance, the view poses are adjusted according to the new bin location, whereas if the pose difference is too large, the process aborts and requires a new round of offline simulation. The reason for this latter behaviour is due to the fact that when the pose difference is large, the optimal path calculated offline may have become sub-optimal since it was originally generated considering the bin obstacle in a different location with respect to the robotic arm base.

In short, in the offline mode, the view poses are generated, filtered and optimized, and the shortest path passing by the resulting ones is calculated. In the online mode, instead, the view poses are adjusted, and the robot moves to each view pose, possibly in the optimal order, to shoot a picture of the bin surface and, if desired, request the classification of the visible surface. We better describe the parameters involved and discuss the results in Sec. 4.2.2.1-4.2.2.2.

In both cases, one must define whether the robot should scan the internal or external surface before launching the application. According to the selected option, a different bin mesh file is loaded. The CAD model for the external surface scanning, in fact, is artificially closed so that there is no internal surface included in the computation (Fig. 4.12a). The closing top and bottom circular surfaces are removed at run time to ignore those artificial portions. Instead, the CAD model of the internal surface is the original open mesh, and only the internal side is considered by removing the external one at run time (Fig. 4.12b).



(a) Used for external surface scanning.

(b) Used for internal surface scanning.

Figure 4.12: Bin meshes.

Among the settings, there is also the homogeneous transformation matrix which expresses the marker pose with respect to the bin reference frame. The transformation is used to correctly define the position of the robot relative to the bin after the marker pose has been estimated (or loaded) in robot coordinates.

4.2.2.1 Offline

The first viewpoint computation, which is the one independent of robot kinematics and dimensions, depends on a few user-defined parameters used at the different steps described in Sect. 4.2.1.1. For convenience, we report them here with their practical implications:

1. the point-cloud sub-sampling step takes the given percentage value p to sample a subset of uniformly distributed seed points from the set of points obtained through direct mesh-to-point conversion; the larger the value, the more seed points (and therefore seed faces and viewpoints) will be considered; if on the one hand, more viewpoints ensure better coverage of the surface, on the other the computational time significantly increases, especially during the optimal path search; a value of 40% proved to provide good coverage and a reasonable processing time for the meshes at disposal;

2. for each seed point/face, many view-poses are generated as the number D of given distance values introduced in Sect. 4.2.1.1 but in the end, only one will be kept, i.e. the first valid one in the list; the reason behind the choice of having multiple (e.g. D) viewpoints along the same direction is to increase the chance that that direction is used in spite of collisions or unreachability of the target

pose; the values d_i included in this list should be within the boundaries defined by the range of view;

3. the filtering steps may apply up to four filters, discussed in Sec. 4.2.1.1; we report hereafter a few practical implications regarding the implementation of some of them:

    - *height check* : to implement the impossibility of the camera of being underground, all viewpoints whose origin's z-component is below mesh bottom-most coordinate minus the maximum dimension[3] of the camera mesh (if given) are removed;
    - *distance check* : the reference point for the distance calculation is the origin of the robot (base) frame expressed in the bin coordinates system;
    - *collision check* : from a physic point of view, the view-pose coincides with the optical frame of the camera; this filter may use the homogeneous transformation matrix that expresses the optical frame with respect to the camera/tool mesh frame; if not given, the camera/tool mesh frame will be coincident with the optical frame.

The second round of filtering, which is significantly more computationally expensive than the first one, tests each remaining view pose by trying to plan a trajectory with the robot and the updated planning scene that includes the bin as an obstacle. During this phase, the robot tries to align the optical frame of the RealSense camera mounted on its wrist to each view pose. Since we are not interested in the camera's x and y-axis orientation, the planner is free to reach that pose with any rotation around the z-axis. If the pose is reached with a different xy-orientation than the ideal one, the view pose is updated.

The viewpoint selection process may begin with this new set of filtered and reachable view poses. The first two parameters involved at this stage are the Distance of View (DoV), which defines the visible range of the sensor in terms of depth, and the maximum glance angle ($\alpha_{g,\max}$), which defines the maximum inclination with respect to a surface that a view ray may have to see it properly. With this information, it is possible to compute the measurability matrix used by the GA method. With the parameter p, the user may define the percentage of viewpoints that should be removed by the SA method from the ones resulting from the previous GA method. If the value is set to 0, the SA optimization is skipped. From experience, when the viewpoints obtained with the GA method are a passable number (e.g. ≥ 15), a reduction of 15% usually entails a loss in the visible surface of less than 1%, which is a reasonable trade-off.

The last step consists of sorting the viewpoints according to the best path. This is also the most time-consuming stage since it must test all motions from pose i to pose j, with $i = 0, \ldots, N_{opt} - 1$, and $j = 0, \ldots, N_{opt} - 1$, being N_{opt} the number of selected viewpoints by the previous optimization steps.

In Table 4.1, the values of the parameters used in this application are listed, whereas in Tables 4.2,4.3, the results of a few rounds of this process are reported.

By looking at Table 4.2, it can be noticed how the first round of filtering plays a major role in the decimation of viewpoints, especially in the external-surface case with an

[3] When dealing with meshes, by *dimensions* we refer to the sizes of the 3D object in the Cartesian space, often addressed as height, width, and depth.

Parameter	**Symbol**	**Value(s)**
distance-of-view range	DoV	[0.175 : 0.6] [m]
cobot max reach	$\overline{r}$	1.3 [m]
camera mesh frame to optical frame transformation	$^{opt}\mathbf{H}_{cam}$	$\begin{bmatrix} 1 & 0 & 0 & 0.118 \\ 0 & 1 & 0 & 0.058 \\ 0 & 0 & 1 & -0.035 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
bin mesh frame to ChArUko marker frame transformation	$^{tag}\mathbf{H}_{bin}$	$\begin{bmatrix} 0.9997 & -0.0233 & 0.0105 & 0.0277 \\ -0.0095 & 0.0391 & 0.9992 & 0.1491 \\ -0.0237 & -0.999 & 0.0388 & -0.3838 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
maximum glance angle	$\alpha_{g,\max}$	60°
SA reduction percentage	p_{SA}	15%
distances from seed point - int	d_0, d_1, d_2	$[0.4, 0.55, 0.2]$ [m]
distances from seed point - ext	d_0, d_1	$[0.3, 0.4]$ [m]

Table 4.1: Bin scanning parameters. The translation block in the homogeneous transformations is expressed in meters.

Surface side	Seeds % (p)	VP	VP after filtering	VP after reachability check	GA VP	SA VP (% visible area loss)
external	1	65	22	13	12	10 (-1.76%)
external	5	327	105	68	31	26 (-0.19%)
external	10	655	195	137	36	30 (-0.1%)
external	40	2469	714	484	52	44 (-0.11%)
external	70	3891	1097	749	55	46 (-0.38%)
internal	1	8	6	4	4	3 (-2.73%)
internal	5	44	41	20	13	11 (-0.09%)
internal	10	89	80	34	15	12 (-0.67%)
internal	40	323	288	137	15	12 (-0.4%)
internal	70	500	440	189	17	14 (-0.52%)
internal	100	599	518	213	20	17 (-0.38%)

Table 4.2: Viewpoints decimation with different seeds percentage values for internal and external surface and SA reduction of 15%.

average of -69% against the -13% of the internal-surface case. This is mainly due to the fact that a great portion of the external surface is out of reach when approaching the bin from one side, and the presence of the external pillars often causes a collision between the camera mesh and the bin mesh. The behaviour is the opposite when applying the reachability check. In fact, an average further reduction of -52% is observable for the internal case, against the lower -34% for the external one. This, in turn, is due to the higher difficulty for the robotic arm to find a suitable configuration that allows it to position itself inside the bin by approaching from the top without redundant DoF. Another interesting piece of information is related to the amount of viewpoints

Surface side	Seeds %	VP generation and filtering	Reachability check	VP selection	Distance matrix	ACO
external	1	0.4s	33s	0.1s	54.3s	0.1s
external	5	2s	2m 34.4s	0.6s	6m 31s	0.2s
external	10	4.2s	4m 15.8s	0.3s	8m 41.6s	0.4s
external	40	13.8s	15m 56s	1.3s	19m 22.3s	0.8s
external	70	22.9s	23m 58.6s	2.6s	24m 48.1s	0.9s
internal	1	0.1s	14.6s	0.1s	2.3s	-
internal	5	0.3s	2m 10.4s	0.2s	1m 8.1s	0.1s
internal	10	0.6s	4m 32.5s	0.3s	1m 33.7s	0.1s
internal	40	1.7s	15m 19.2s	0.1s	1m 52.4s	0.1s
internal	70	2.6s	24m 29.5s	0.6s	2m 9.7s	0.1s
internal	100	3s	29m 22.5s	0.1s	3m 50s	0.1s

Table 4.3: Execution times with different seeds percentage values for internal and external surface and SA reduction of 15%.

selected by the GA method. Noticeably, the actual number of necessary viewpoints to see the same surface as with the initial ones does not follow the same proportional trend, meaning that increasing the initial amount of viewpoints does not add much information (i.e. visible area) to the whole process. This is more evident in the external surface case, shown in Fig. 4.13a, than for the internal one, shown in Fig. 4.13b. On the contrary, by looking at Table 4.3, computational time linearly grows as p increases, with a major contribution of the reachability check and the distance matrix computation steps, as expected and shown in Fig. 4.14a-4.14b. As a conclusion, from experience, we found that a seed percentage value p between 10 to 40 is a good trade-off between area coverage and computational time. Nonetheless, if the objective is to maximize the visible area disregarding the time and effort, all available seeds should be taken (i.e. $p = 100\%$) as initial attempt.

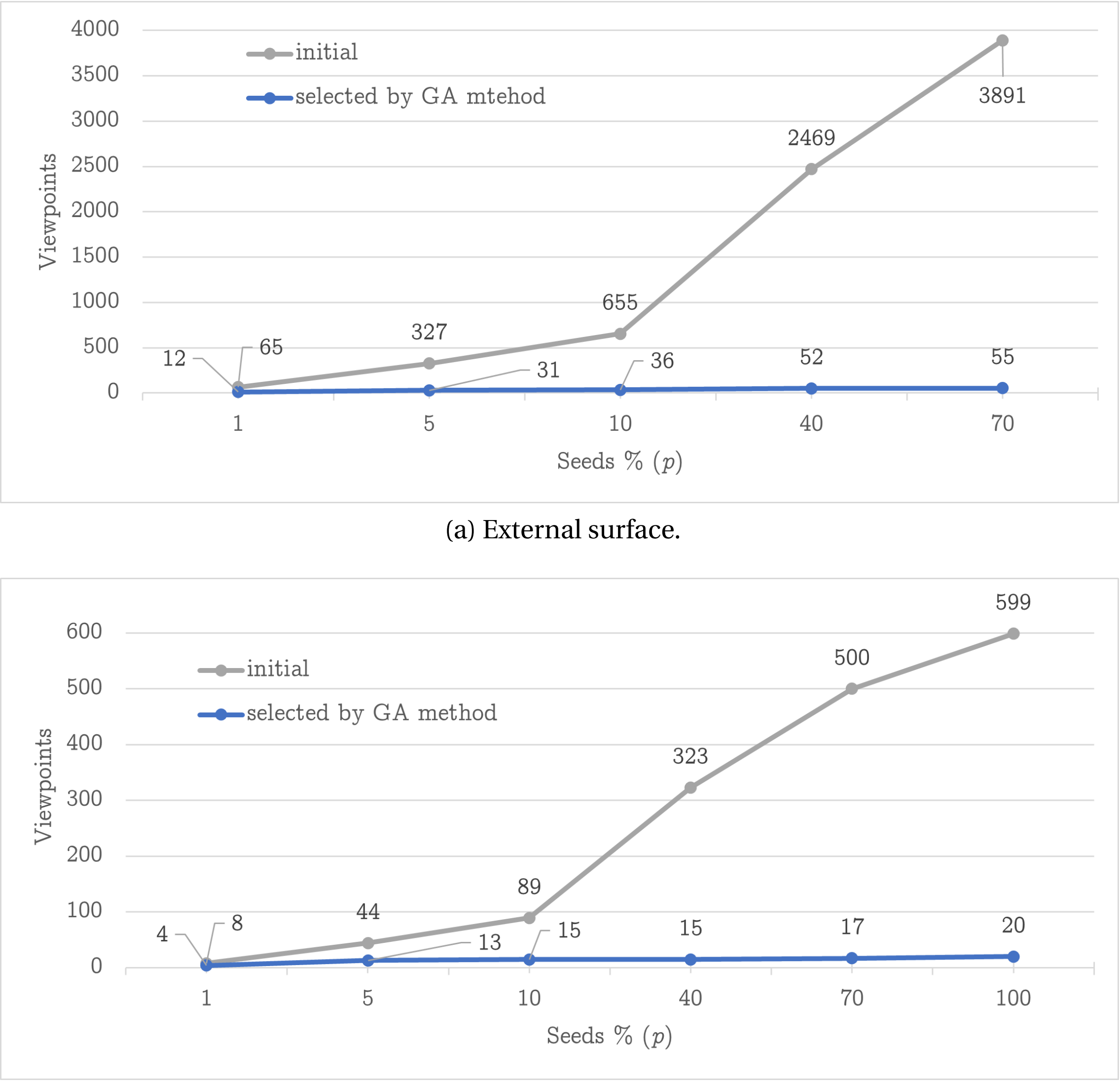
The last piece of data worth analysing is the amount of area loss when applying the SA method with a reduction percentage of 15%. Except for the case where $p = 1\%$, the average loss is below 0.5% for both surface sides. A loss above 1% is only obtained when the amount of starting viewpoints (obtained from GA method) is already small with respect to the total area size. In this case, even removing a single viewpoint can lead to a significant loss in the visible area.

4.2.2.2 Online

The online mode can be used for two different purposes: data collection or evaluation.

In the first case, all reachable viewpoints are used, ignoring the selection and sorting steps in order to maximize the number of images that can be autonomously collected for the training of the classifier. To this end, a perturbation to the target pose may be added, if desired, to introduce some noise and help the training process by increasing the variance of the data.

In the evaluation case, only the selected viewpoints are used instead, and the sorted order may be followed if desired. In this scenario, no perturbations are applied. If the evaluation is required, after stopping at a view pose, a request is sent to the bin classi-



(a) External surface.

(b) Internal surface.

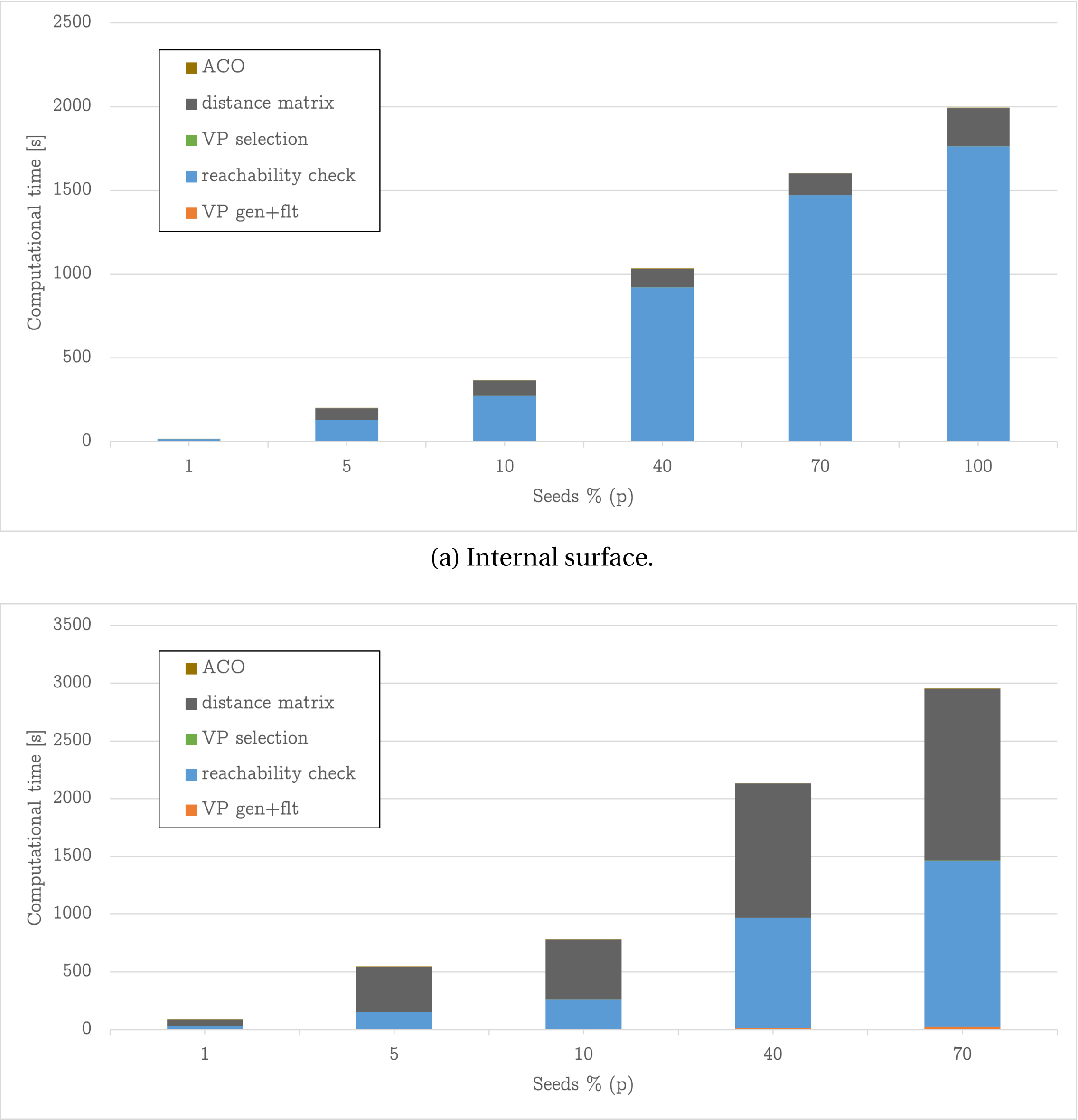
Figure 4.13: Viewpoints reduction against different seeds %.

fier node, and the results are published on the network through designated *ROS topics*. Before the request is emitted, the procedure waits 1 second to let any arm oscillation disappear and let the image be in focus.

4.3 Bin-surface classification

In machine learning, *classification* refers to a predictive modelling problem where a class label is guessed for a given example of input data. Practical examples can be the classification of an e-mail as spam or not or, given a hand-written letter, classify which letter it represents. There are four main types of classification:

- *binary classification*: it refers to predicting one of two classes (e.g. spam vs not spam, male vs female, dark vs light, etc.);
- *multi-class classification*: it refers to predicting one of many classes (e.g. alphabet letter, dog breed, species of plants, etc.);



(a) Internal surface.

(b) External surface.

Figure 4.14: Computational times against different seeds %.

- *multi-label classification*: it refers to predicting one or more classes for each sample (e.g. a picture of a horse could be classified as "animal", "horse", and "Albanian horse" at the same time);

- *imbalanced classification*: it refers to classification tasks where the distribution of examples across the classes is not equal; this is typically the case for anomaly detection, which often belongs to uneven binary classification, given the usual low occurrences of the anomaly with respect to the nominal case in the training samples.

The use case at hand falls under the binary classifications, given the fact that the goal is to distinguish a clean surface from a dirty one and the training data set is evenly distributed. The samples, in this case, are represented by images of a portion of the bin surface taken at an suitable distance in the visible range of the 2D RGB camera (e.g. RealSense D435). The images may include some unrelated background; they are in focus and are cropped and resized to be independent of the original resolution.

4.3.1 Methodology

In simple words, *Artificial Intelligence (AI)* is a science whose goal is to make machines think and act like human beings. *Machine Learning (ML)*, in turn, is a subset of AI focusing on allowing computers to perform tasks without the need for explicit programming [66].

Suppose to have a set of input-output pairs, called *training set*, $\langle x_i, y_i \rangle$; the problem consists in guessing the best map $x_i \mapsto y_i$. In ML, such a problem is described with a *model* that depends on some parameters Θ. The model can be either chosen from a class of parametric functions (regressors, polynomials, etc.) or manually engineered by a specialized programmer. A *loss function* is defined to compare the result of the model (i.e. *predictions*, $\tilde{y}_i$) fed with the input x_i with the expected measured/experimental values, y_i. Lastly, the parameters Θ are optimized (or fit) to reduce the loss to a minimum through an iterative technique, such as gradient descent. ML problems are, as a matter of fact, optimization problems where the solution is not given in an analytical form, often because a closed-form solution does not exist at all.

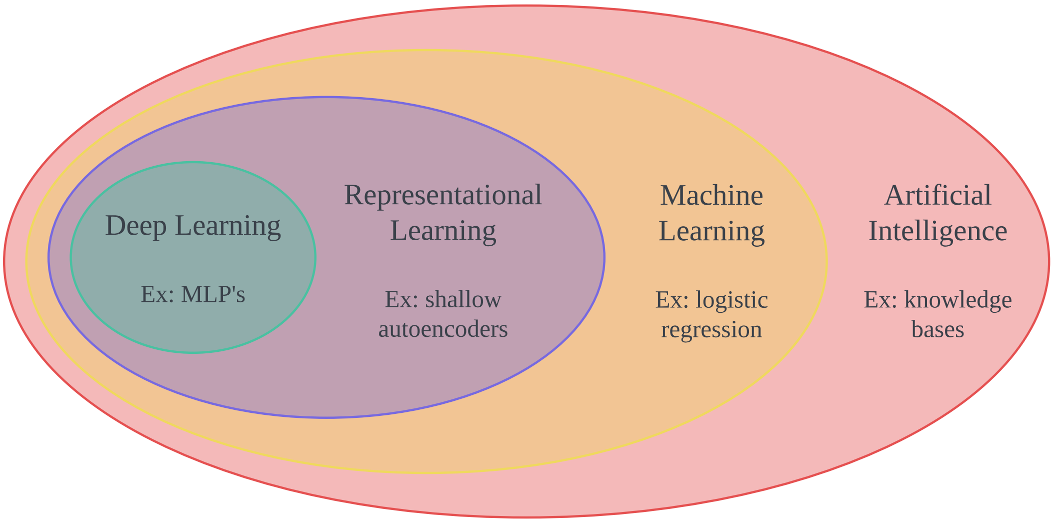


Figure 4.15: Venn diagram of AI disciplines.

Deep Learning (DL), at last, is a branch of ML (Fig. 4.15-4.16) that may target all problems suitable to ML but it truly excels in those involving myriads of features such as images, speech and text processing [67]. DL modelling introduces an extremely sophisticated approach based on complex, multi-layered *Neural Network (NN)*, built to allow data to move through nodes (like neurons) in highly connected ways. The term "deep" comes from the depth of the networks, that is, the long chain of neurons that goes from the input to the output through many intermediate layers. As a consequence, "deep" features of the data are progressively extracted from other features, leading to a level of detail which is hardly attainable by standard ML approaches. The result is a non-linear transformation of the input data through an increasingly abstract model that, thanks to this peculiarity, can adapt to a wide range of problems. The curious reader may find a brief and more detailed analysis of NN in Appendix D.

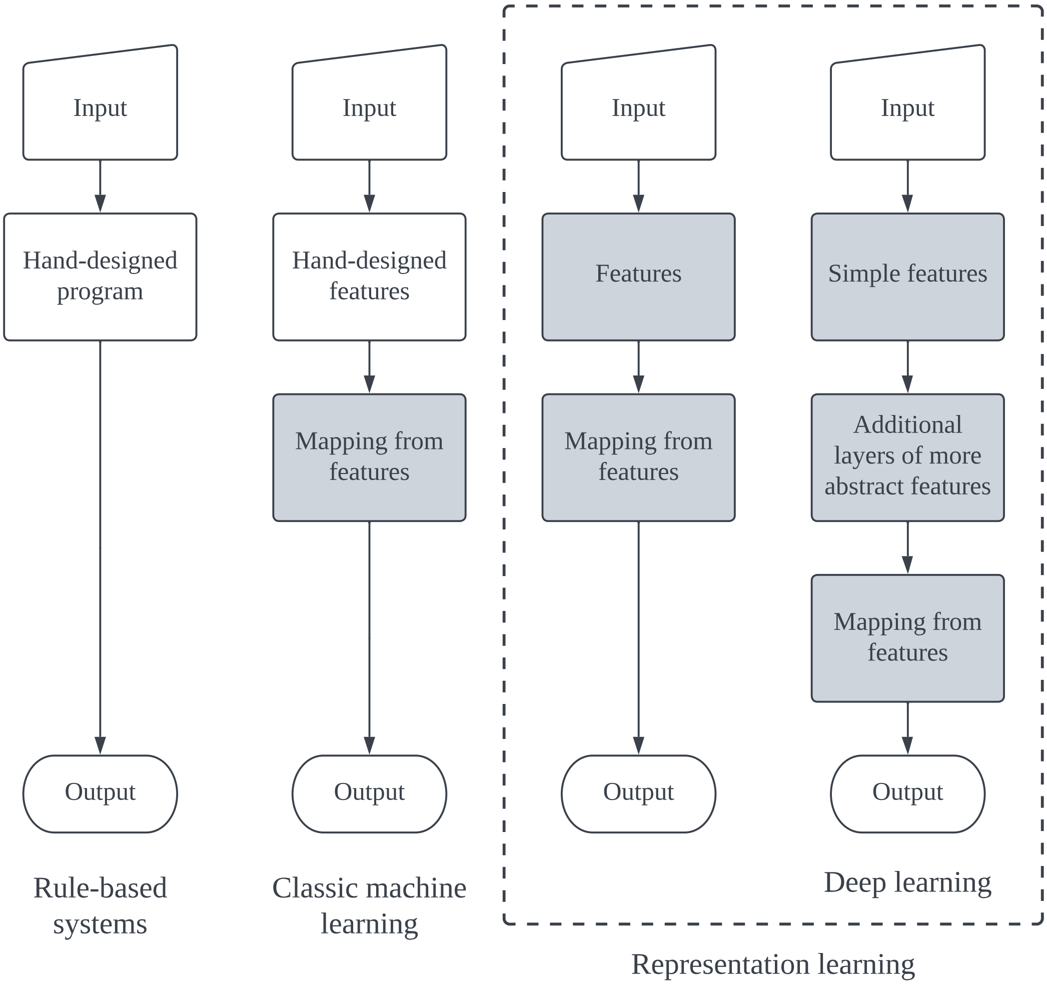


Figure 4.16: Flowchart of different AI parts related to different disciplines. Shaded boxes indicate components that are able to learn from data.

The desired classification problem is particularly suitable for resorting to *Convolutional Neural Network*. This technique, in fact, provided ground-breaking results in the field of image analysis and Computer Vision (CV), outclassing standard CV techniques in terms of efficiency, but especially in terms of robustness and flexibility.

4.3.1.1 A brief introduction about Convolutional Neural Networks

A *Convolutional Neural Network (CNN)* is a particular type of *Deep Neural Network (DNN)*; thus, a DL algorithm that employs *convolutions* in at least one layer [68] [69]. It proved to be extremely suited for image processing thanks to its ability to assign importance to different features and/or objects in this particular type of input and in a very efficient way [70].

The idea of the general architecture of CNN comes from the analogy with the connectivity pattern of neurons in the human brain and how the *visual cortex* is organized. Each individual neuron reacts to stimuli only in a restricted visual field region, also called *receptive field*. A set of such overlapping fields eventually covers the entire visual area.

A CNN can capture both spatial and temporal dependencies in an image through the consecutive application of suitable filters. Differently from primitive methods, where filters are hand-engineered, CNNs are able to learn them by training on labelled data autonomously, that is, in this case, images with an associated ground-truth description/label. This technique, therefore, falls under the realm of *data-driven supervised learning*, where the goodness, distribution and availability of labelled data play a key role in the robustness and efficacy of this method.

Starting from a multi-channel image (RGB, HSV, greyscale, etc.), the goal of the CNN is to compress the image into a form that is easier to process while maintaining the features critical for obtaining a good prediction. To do so, the key element of the CNN is the *convolutional kernel*, a $m \times m \times d$ matrix $\in \mathbb{R}^3$, where m is the size of the kernel and d is its depth. A multi-channel image is nothing but a matrix $\in \mathbb{R}^3$ of $w \times h \times c$ dimension, where w is the width in pixels, h is the height in pixels, and c is the number of channels depending on the colour space. The kernel must have the same depth as the number of channels of the input matrix, that is, $d = c$. The convolution of the input image with the kernel can be obtained by sliding the filter over the image and computing the dot product of each receptive field of the image with the kernel. The receptive field, in this case, consists in the portion of the input matrix interested by the kernel at each iteration, and the dot product is the sum of the element-wise multiplication between the receptive field and the kernel matrices. This computes one number which populates a single-value output image, also called *output activation*. A visual example of this process is illustrated in Fig. 4.17.

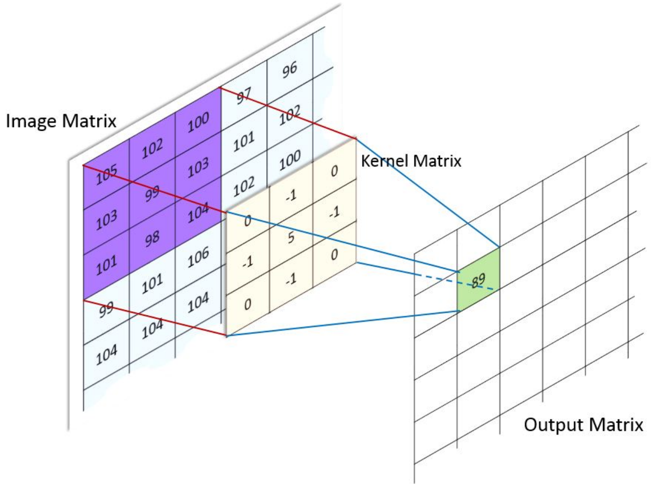


Figure 4.17: Example of a single step of a matrix 3 × 3 × 1 kernel convolution. The purple area is the receptive field for the considered position. Image taken from online GitHub repository https://github.com/ashushekar/image-convolution-from-scratch.

As usual in DNN, affine functions also include a bias; therefore, a scalar value is associated with each kernel filter, as shown in Fig. 4.18.

N kernels can be stacked together to produce N output activations from the same input matrix. The set of kernels + biases composes a so-called *convolutional layer*, illustrated in Fig. 4.19. In CNNs, convolutional layers are applied in sequence; therefore, the output activations generated by one layer become the input matrix of the next one. Playing on *stride*, *padding* and dimensions make it possible to build a suitable architecture that extrapolates the most important features from an image, starting from low-level characteristics such as edges and corners, up to more high-level clustering

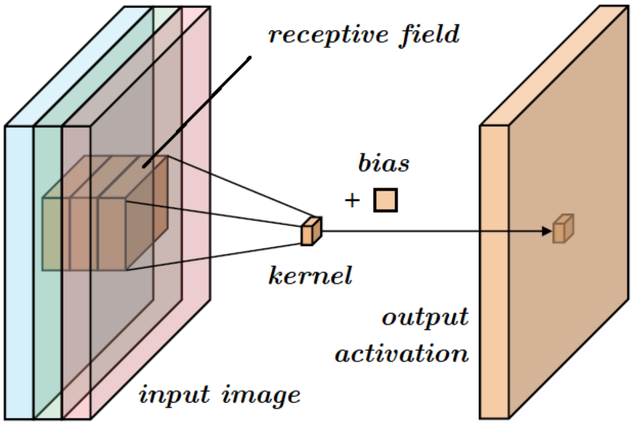


Figure 4.18: Example of an affine convolutional filter.

and patch recognition.

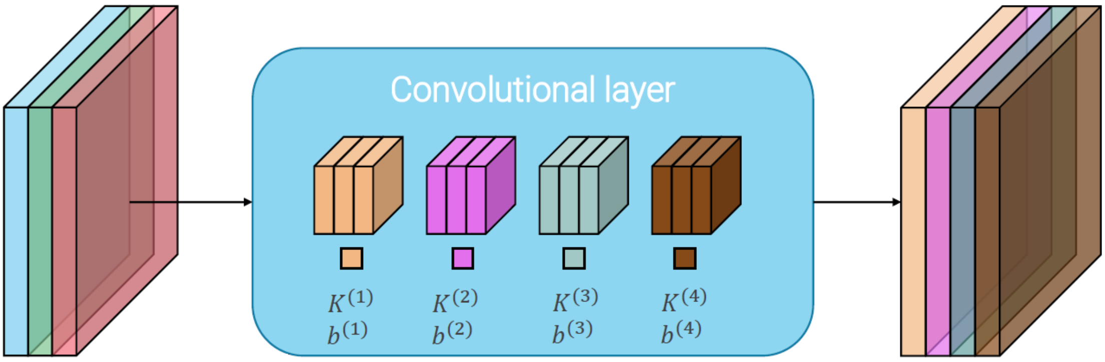


Figure 4.19: Example of a convolutional layer.

Besides convolutional layers, CNNs are a composition of *non-linear activations, pooling layers, batch-normalization layers* and *fully connected layers*. With reference to Fig. 4.20, convolutional layers are used in the first section of the DNN and compose the so-called *feature extractor*, which may be generic and suitable for many applications. Instead, the second and last portion is characterized by fully connected layers that use a flattened version of the resulting compression given by the feature extractor and can be used to define a specific classification problem thanks to a final *softmax* layer. Hence, it is also called *classifier*.

The *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* compares and evaluates algorithms for object detection and image classification at large scale, becoming a strong motivation to researchers that contribute to the progress of object detection and classification across a wider variety of categories since 2010. In Fig. 4.21, the classification for the winners in the challenge is shown and demonstrates how *AlexNet* [72], which was the first approach that employed CNNs, made a great leap down in the error rate. Based on visualization and ablation studies, Zeiler and Fergus [67] found out that aggressive stride[4] and large filter size in the first layer result in dead filters and missing frequencies in the first layer filters and aliasing artefacts in the second layer activations. To counteract these problems, in 2013, with their *ZFNet*,

[4] *Stride* is the quantity that defines how far the convolutional filter moves in every step along one direction.

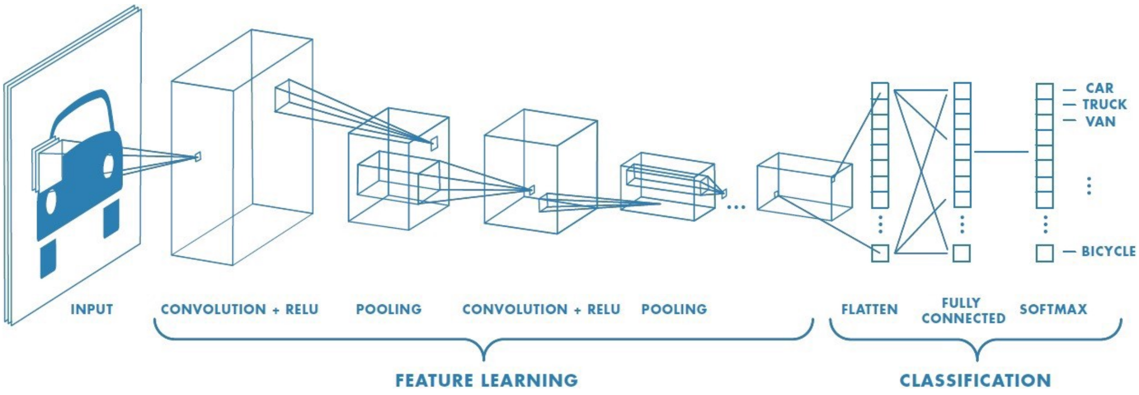


Figure 4.20: Generic CNN architecture. Source: A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.

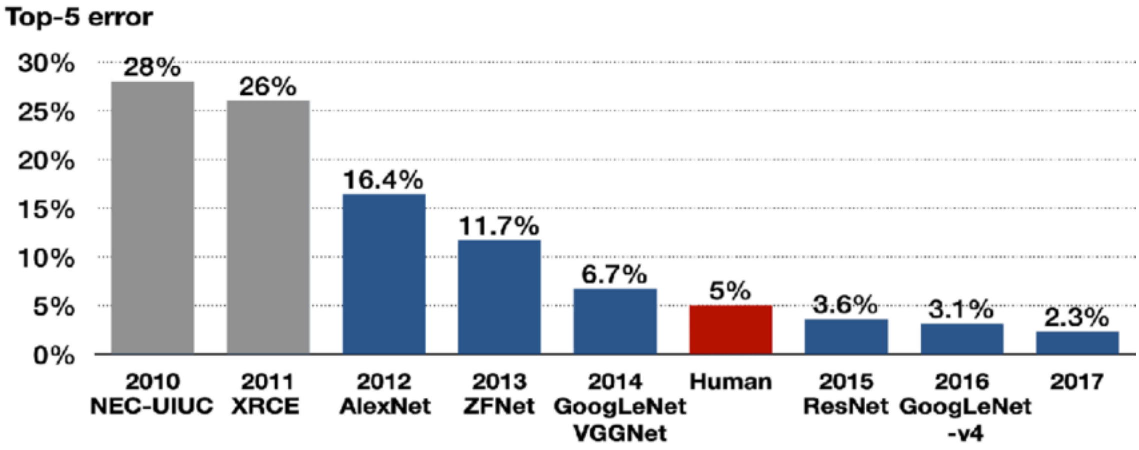


Figure 4.21: Algorithms that won the ILSVRC in 2010-2017. The top-5 error refers to the probability that all top-5 classifications proposed by the algorithm for the image are wrong. The algorithms with the blue graph are convolutional neural networks, whereas the ones with the grey graph are feature engineered. Picture from [71].

they propose to use 7 × 7 convolutions with stride 2 in the first layer and stride 2 also in the second 5 × 5 convolutional layer. In that same year, the second place went to Simonyan and Zisserman [73] for their *VGG* network, which explored the effectiveness of simple design by introducing the concept of repetition of *stages*, an almost-fixed combination of layers made of 3 × 3 convolutions and 2 × 2 max-pooling. One stage is built so that it has the same receptive field of larger convolutions but requires fewer parameters and computations and introduces more non-linearities (which is desired in DNNs). In 2014, Szegedy et al. achieved the first position with *GoogleLeNet* [74] "by a carefully crafted design that hat allows for increasing the depth and width of the network while keeping the computational budget constant". This was done by using *stem layers* to aggressively down-sample inputs, introducing the *inception module* and the *global average pooling classifier* to replace the last fully connected layers. Inspired by VGG regular design, *ResNet* [75] tackles the problem of the inability of VGG to learn the identity functions by introducing the *residual blocks*, implemented by adding *skip connection* skipping two convolutional layers, as shown in Fig. 4.22. Moreover, it uses the stem layer and global average pooling as in GoogleLeNet. Following the VGG convention, by the *X* in ResNet-*X*, it is indicated the number of layers with learnable parameters. This is also the first CNN to achieve predictions above the human level.

4.3.1.2 ResNet-based application

Given the fact that subsequent works after ResNet are just more complicated and power-consuming combinations of previous CNNs with a negligible gain in performance, *ResNet18* is chosen as the backbone for the bin surface binary classifier for its availability as open-source software and excellent performance. Thanks to the so-called *transfer learning* technique, it is, in fact, possible to exploit the pre-training of this sophisticated CNN over millions of image samples and use the small amount of application-specific pictures to replace the classifier section and finely tune the feature extractor layers. This method is based on the idea that, especially when dealing with images, a common processing part extracts high-level, generic features from a picture. These features can then be suitably combined to solve a specific task, for instance, a binary surface classification (e.g. clean vs dirty).

Hence, the input of our customized, ResNet-based network is an RGB image framing a portion of the bin surface, and the output is a label indicating whether the surface is clean or not.

Because there is a substantial difference between the surface inside the bin, which is polished and highly reflective, and the one outside, which is instead raw and opaque, it was opted for a two-element ensemble of networks: one for the former and one for the latter. The two networks are, thus, trained separately and used according to the side of the bin which is being scanned.

Another advantage of this method is the possibility to easily re-train the network whenever there are changes in the specifications, such as a different powder or a different bin surface are used. This characteristic can also be exploited to re-train the network with a larger and/or better data set.

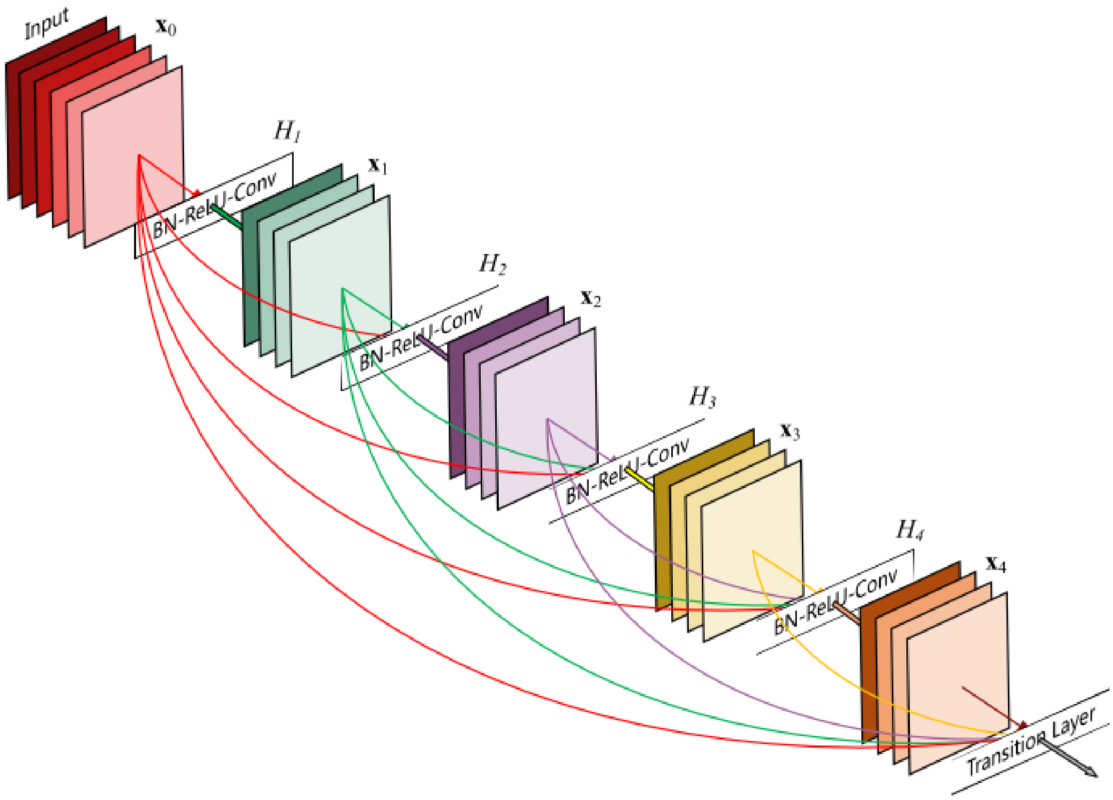


Figure 4.22: ResNet architecture overview.

4.3.2 Implementation and results

The *ROS package* developed for the SENECA project includes two main items: a standalone script to arrange the data set and train the networks (one per surface type) and a *ROS node* that uses the trained model, connects to a camera stream, and can predict the class of the latest grabbed image.

The CNN implementation is made using PyTorch [46], an open-source ML framework that accelerates the path from research prototyping to production deployment by providing blocks, models, pre-trained parameters and functionalities related to ML and DL algorithms.

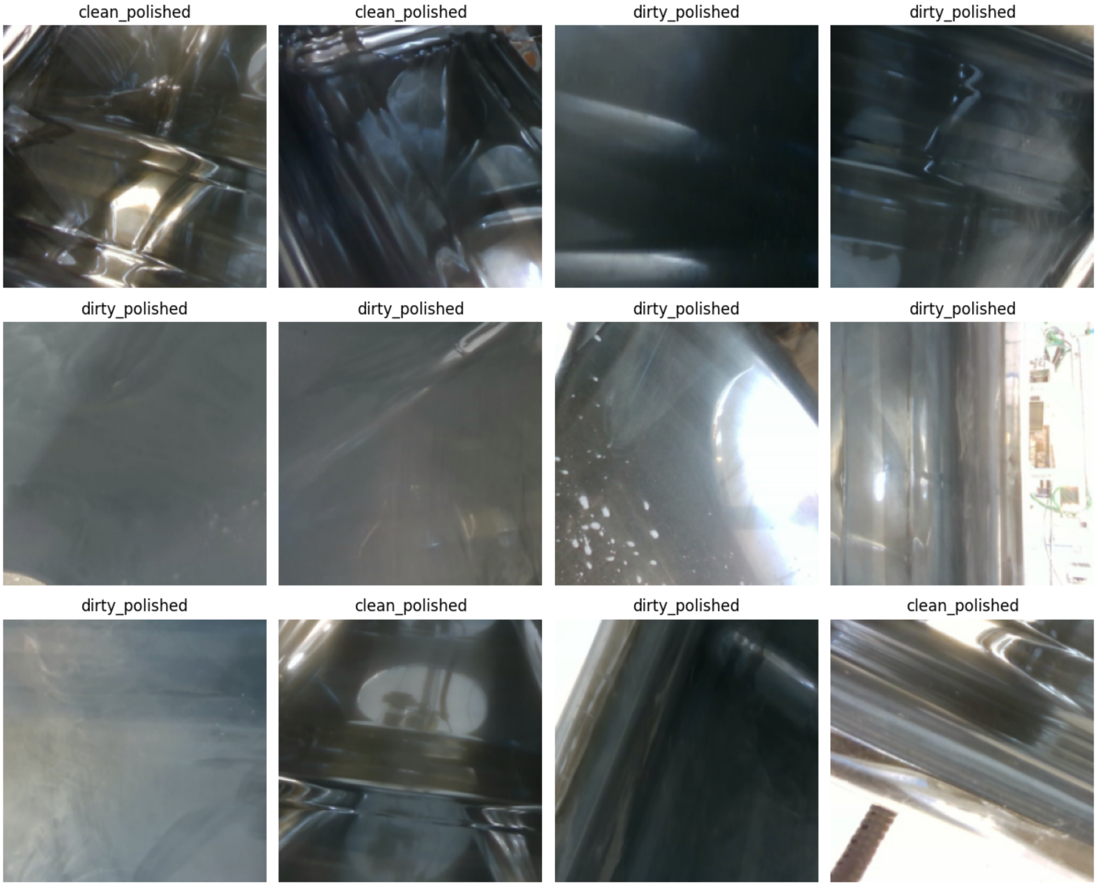


Figure 4.23: Random subset of sample training images for the internal surface classifier.

In Fig. 4.23, we show a random subset of images with the corresponding ground truth (i.e. correct label) used to train the network tailored to the internal surface. Similarly, in Fig. 4.24, a subset used to train the one for the external surface is shown. In Fig. 4.25, the distribution of training, validating and test set is displayed for both cases. As one can see, around 500 samples per class were automatically collected for the internal surface case, whereas around 1000 samples were collected for the external surface case. The reason behind this discrepancy is due to the greater difficulty for the robotic arm to access the internal side of the bin to take pictures during the data collection phase. Moreover, the total area of the internal surface is smaller than the external one, leading to a smaller number of seed points and, thus, of viewpoints.

The hyper-parameters listed in Tables 4.4 were used to train the model in two consecutive steps: first, only the classifier layers are trained by freezing the parameters of the feature extractor; then, at the fine-tuning phase, the whole network is trained by

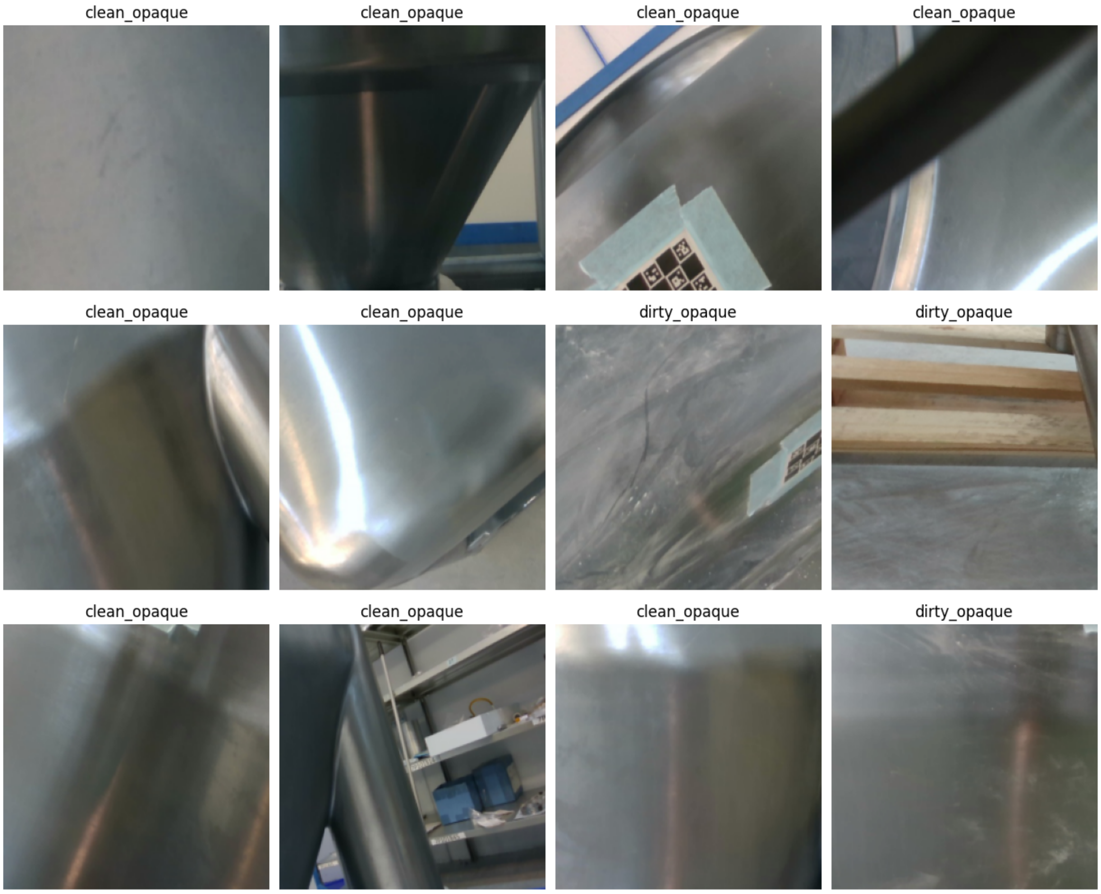


Figure 4.24: Random subset of sample training images for the external surface classifier.

unfreezing them and by using a lower learning rate but a larger number of epochs. More details about the terminology related to the implementation of a NN can be found in App. D.

Name	Value(s)
learning rate	0.001
num. of epochs	30
image size	244 × 224 [pixels]
batch size	32
momentum	0.9
weight decay	0.0001

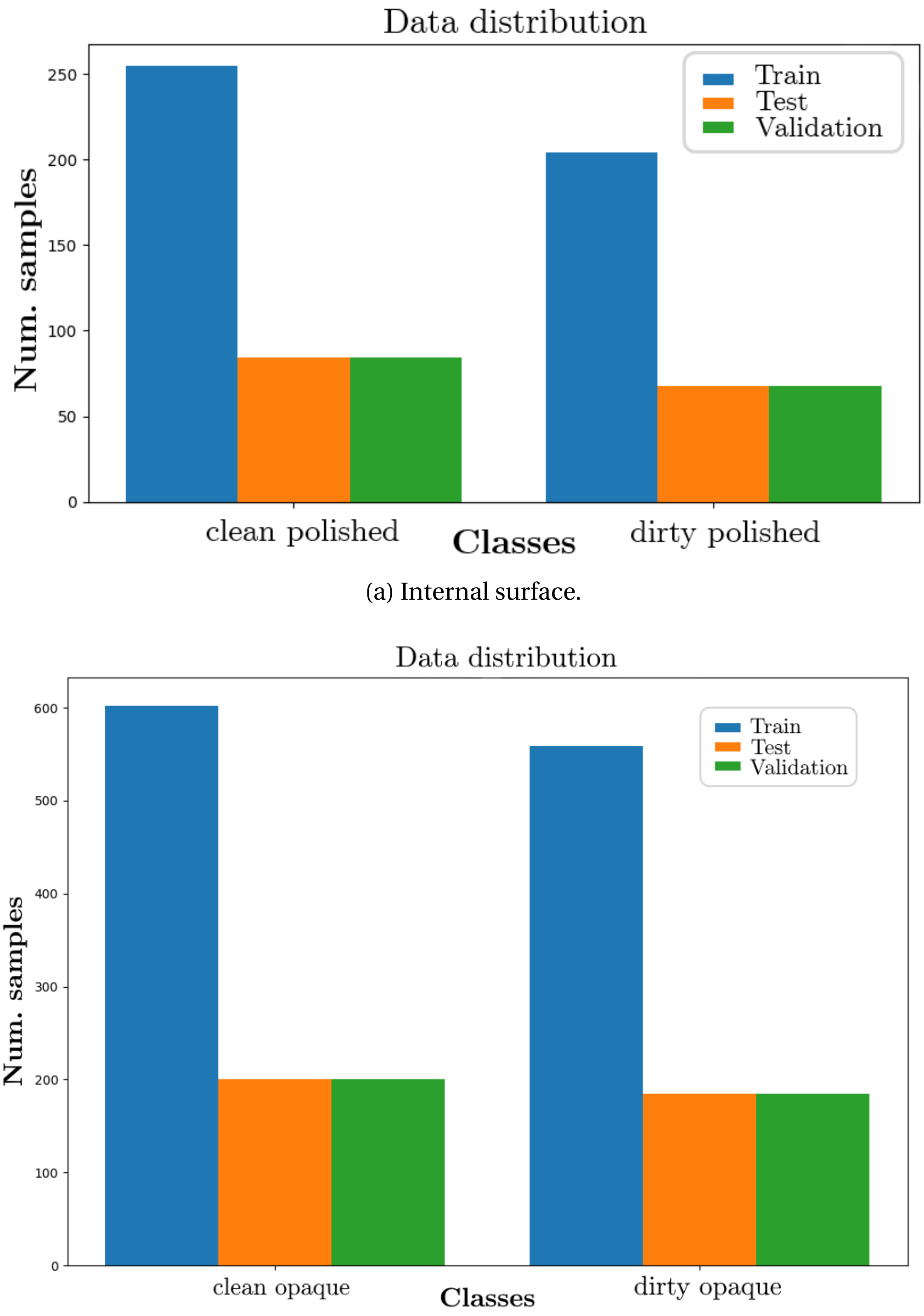
(a) Classifier.

Name	Value(s)
learning rate	0.0001
num. of epochs	100
image size	244 × 224 [pixels]
batch size	32
momentum	0.9
weight decay	0.0001

(b) Fine tuning.

Table 4.4: Hyper-parameters used at training time.

To augment training data, images are randomly cropped and/or horizontally flipped, whereas to enhance batch normalization, images are normalized using mean values and standard deviations for each one of the three colour channels, directly calculated on the training + validation image sets. *Stochastic gradient descent* is used as optimizer with *momentum* equal to 0.9 and *weight decay* equal to 0.0001. A *learning rate scheduler*, instead, decays the *learning rate* of each parameter group by 0.1 every 5 epochs. The DNN-related terminology is briefly explained in App. D.



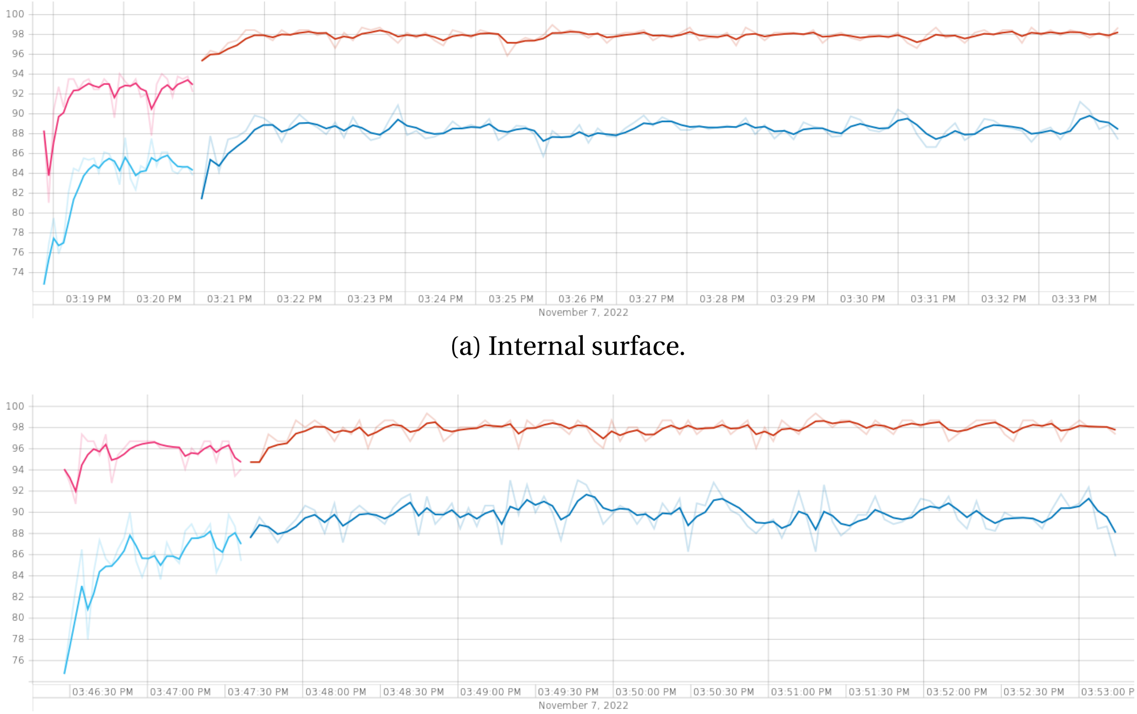
(a) Internal surface.

(b) External surface.

Figure 4.25: Data distribution.

Finally, the accuracy for both classifier's training and fine-tuning are shown in Fig. 4.26 for the train and validation + test set.

Despite the short time dedicated to the tuning of the hyper-parameters and the relatively low amount of training data, results show accuracy on the test set around 98% for both scenarios. This excellent outcome, however, must be taken with care since the testing data were collected under the same conditions as the training set. This means that the combination of light conditions, type and status of the inspected surface and the camera itself used to collect testing data is not significantly different from the one used to collect training data. The main difference is indeed in the different framing perspectives, which, however, introduce a variability not sufficient to truly verify the robustness of the model. Hence, a larger variance in the working scenarios would be necessary to test the quality of the model.



(a) Internal surface.

(b) External surface.

Figure 4.26: Accuracy plot for bin cleanliness classifier. Magenta and light blue lines represent, respectively, the accuracy of training data and validation data during the classifier's training. Orange and blue lines represent, respectively, the accuracy of training data and validation data during fine-tuning.

Chapter 5

Conclusions

In this Thesis, we presented two industrial applications that employed the same Autonomous Mobile Robot (AMR) to solve different tasks in distinct scenarios.

The robotic platform was originally developed within the context of the ROSSINI EU project, a four-year-long investment whose ultimate goal was to demonstrate a modular and scalable platform for the integration of human-centred robotic technologies in industrial production environments. In particular, this work contributed to the IMA use case, one of the three industrial scenarios targeted to prove the inherent versatility of the ROSSINI technologies and framework. The goal was the automatic loading and unloading of raw-material reels to an automatic packaging machine, a task already objective of other previous projects, namely the EuRoC and MaXima projects. The latter was also briefly described in the Thesis to highlight the technology evolution, keeping the successful key elements and improving the ones that showed more critical issues. The main achievements within the ROSSINI scope were related to the ROS-based software architecture of the robotic system and implementation of the aforementioned routines.

A noteworthy mention must be made to a robust vision-based procedure for autonomous change of reel with the mobile robot. The envisioned solution was implemented to be easy to share, document, maintain and be deployed on an Industrial Personal Computer with limited power resources, such as the one installed onto the ROSSINI AMR. Three methods for detecting a single ellipse out of a picture framing the reel core were proposed (i.e. contour, RBrown, and Hough transform), and the results coming from an extensive experimental campaign showed how, following an optimal selection policy, the success rate touched 99.6%. We also indicated how the different photo-metric characteristics of the reel affect the efficacy of each method. This qualitative assessment can be a useful contribution to the generic problem of estimating the position and orientation of a circular object in any industrial scenario under specific circumstances without resorting to overly sophisticated techniques and high-power-consuming devices.

With reference to the overall reel-change procedure, the outcome was positively evaluated by the EU commission, who selected the IMA use case for an in-house demonstration. The task was performed in about 5 minutes in the absence of an Safety-rated Monitored Stop triggered by the presence of a human close to the robot. The relatively long execution time was mainly due to the slow speed of the robotic arms' motions, which was kept low to guarantee high safety standards for the human approaching the AMR. Nevertheless, considering the targeted Technology Readiness Level (TRL) 6-7, performance was not the main Key Performance Index (KPI) for the evaluation of the ROSSINI project, which was, in turn, an enhanced Human-Robot Collaboration (HRC) supported by the integration of the RS^4 safety system (ROSSINI Smart and Safe Sensing System, a set of new prototypes for a safety-rated 3D camera to track both the position and the speed of each operator and robot in a monitored shared working area). The following observation can be made from experience and tests:

- the filter-paper-reel change yields a higher success rate with respect to the tag-paper case as discussed in Sec. 3.7.1;
- dynamic trajectory planning of the robotic arms is a true game-changer in the field of object manipulation because it allows operating in unstructured environments up to a certain level; however, it sometimes leads to unfeasible motions due to the lack of clamping hazard check, the poor handling of singularities and varying inertia of the payload; moreover, computed trajectories were often too long, contorted, counter-intuitive and/or unpredictable, which is inconvenient for HRC, where mutual understanding and trust is crucial for acceptance;
- the pipeline for image analysis (e.g. marker detection and reel-core detection) is too convoluted due to the cascade of different client-server structures; if, on the one hand, this helps reuse previous software not directly related to the application, on the other hand, it introduces unnecessary delays and complexity;
- the extraction of the filter-paper-reel core requires a very accurate estimation of the mandrel longitudinal axis, which corresponds to the direction of extraction; if the estimation, which comes from vision, is wrong, the extraction of the core will fail due to the detected high friction forces and the whole mission will be aborted with errors.

In conclusion, with reference to the first application, the proposed ROS-based software architecture and implemented routines played a fundamental role in the success of the overall ROSSINI framework, as expressed by the EU commission who claimed to be positively stroke by the tangible results achieved.

As a second use case, we presented the results of the spin-off project SENECA, aimed at demonstrating the feasibility of a fully automated cleanliness inspection procedure of a pharmaceutical bin through a robotic arm.

To ramp up to the experimental phase, the AMR previously developed for the ROSSINI project was re-used to mimic the approach of a manipulator to a bin with imperfect relative positioning. An additional camera was installed on the robotic arm tool to carry out the classification of the surface status (e.g. clean vs dirty). This distinction was evaluated by a Convolutional Neural Network based on ResNet18 and transfer learning. The viewpoints from which the images are taken are the result of an optimization that, starting from the CAD model of the bin, a ROS-based planning scene and a few parameters, minimizes their amount while maximizing the visible area from the current relative position of the robot with respect to the bin.

The results of this optimization can be used to set up a pipeline which is entirely bin-independent, assuming that the CAD model of each target bin is available. The same procedure may actually be employed also in the generation of the best path to clean the surface by only changing the model that is used to define the frustum of action of the cleaning nozzle. A final tuning phase is still missing, and some code optimization, such as parallelization and GPU employment, could help in speeding up the most demanding steps of the process. However, the most expensive phases can be computed offline for each robot-to-bin relative positioning, relegating the computational efforts to a one-time problem.

As far as the classifier is concerned, preliminary results are very promising, but a complete tuning session is required, and some preprocessing/data augmentation techniques may be included to increase the robustness of the models, which now have an accuracy of 98% on conditioned data.

5.1 Future works

In this last section, we look at several improvements that can be pursued through further developments on the presented subjects.

Recalling the critical issues outlined for the reel-core detection, a possible improvement to speed up the process is porting the same software implementation from Python language to a compiled programming language, such as C++. Additionally, a better sub-sampling strategy and better filters may help reduce the number of iterations executed by the algorithm. With a more performing computing unit, finally, more sophisticated approaches based on the most recent Deep Learning technique may be pursued and lead to similar or better results.

With reference to the overall reel-change procedure, both the robotic and safety system are capable of higher speeds, and further developments and optimization rounds can most probably reduce the total execution time while keeping human co-workers safe. The sub-optimal trajectory generation can be accounted for with a different suitable strategy that takes care of all critical issues mentioned at the beginning of this Chapter. Regarding the image-processing pipeline, given the many *ROS packages* on the topic, a better-integrated solution may be preferred. Due to time constraints, for the moment paper reels can be loaded onto the machine only on the left-side mandrels, but there is no apparent obstacle to cover the right-side mandrels, too; a suitable tuning session and a few additions to the procedure are the only actions needed. The issue related to the friction due to inaccurate axis-direction estimation that invalids the core-removal operation can be handled by a vision-independent approach. In fact, rather than relying on sophisticated methods to guarantee that the vision system works perfectly, a compliant force-based controller could be used to counteract any possible estimation errors. The issue is, as a matter of fact, very close to the peg-in-a-hole problem, to which many solutions are available in the literature [76]. The last expected advancement is linked to the untapped potential of having a dual-armed mobile robot. So far, in fact, only one arm at a time has been used. Still, the benefits of co-manipulation are endless, starting with the unlocked possibility of handling objects that overcome the maximum payload of a single manipulator.

Moving to the future works related to the SENECA project, two lines of further development can be pursued starting from the preliminary results discussed in this Thesis.

On the one hand, there are a few improvements to viewpoint generation that can be carried out, such as:

1. the sub-sampling of the point cloud could be done in order to concentrate more seed points in the most critical areas of the bin, such as corners and junctions, where the dirt tends to accumulate;

2. exploiting this pipeline and working backwards, it is possible to optimize the position of the robot with respect to the bin to maximize the visible area by the sensor/washing nozzle.

3. by placing the robot on the ceiling on top of the centre of the bin, and by adding two degrees of freedom, it could be possible to reach most of the surface of the bin; the additional drives could be a rotational joint around the z-axis of the base of the robot, that would mimic the presence of a rotating platform at the base of the bin, and a linear guide that is able to translate up and down the base of the robot;

4. the Greedy Area method could be upgraded by including in its objective function not only the amount of visible area but also some indices related to its morphology, as proposed in Sect. 3.2.2.2 of [59]; in this way, it would be possible to prioritize those zones where the dirt is likely to accumulate;

5. in the Simulated Annealing method, the same settings proposed by [59] were employed, but a fine-tuning campaign may lead to better results;

6. the same concept may be applied to improve the performance of the Ant Colony Optimization algorithm, possibly finding a relationship between the optimization parameters and the size of the distance matrix (that depends on the selected viewpoints);

7. using a deterministic motion-planning algorithm capable of considering obstacles would make the robot-dependent filtering and the distance-matrix generation more reliable and repeatable; on the other side, given the slight difference between the trajectories computed offline and the ones that must be adjusted online, the use of pre-computed paths might lead to collisions.

With reference to the classifier, instead, possible improvements are:

1. the use of the validation set to tune the initial value of the learning rate, keeping the weight decay to 0;

2. the use of the validation set to tune the value of the weight decay (after 1);

3. the inclusion of dropouts to increase robustness;

4. a different transformation train for data augmentation (colour, affine, etc.);

5. the use a different back-bone Convolutional Neural Network than ResNet18;

6. setting a threshold in the predictions to reduce the number of false positives (e.g. surfaces that are considered clean but are actually dirty) at the expense of increasing false negatives (e.g. surfaces that are considered dirty but are actually clean); in this way, a human-in-the-loop could be in charge of double-checking the detected dirty area and tell the machine whether the prediction is right or wrong; this human feedback may also be included in further training to increase robustness.

An additional feature common to both the viewpoint generation and the classifier would be to match the portion of the visible surface from each viewpoint, calculated from the CAD model, with the predicted label produced by the classifier over the image taken from that same viewpoint. In this way, one could have a precise map of the areas that require more intense cleaning or attention by a human expert.

As one can see, the room for improvement is large. With the appropriate changes, both works have the qualifications to become valid and appealing industrial products in the manufacturing field, thus contributing to reducing the gap between the academy and the real world.

Appendices

Appendix A

Basic notions on the pinhole camera model

Computer Vision is based on the extrapolation of information inherent in the external world through the analysis of images. Hence, a transition from a three-dimensional space to a two-dimensional one is needed. To understand how this transformation occurs and what it entails, we resort to a mathematical model of the camera.

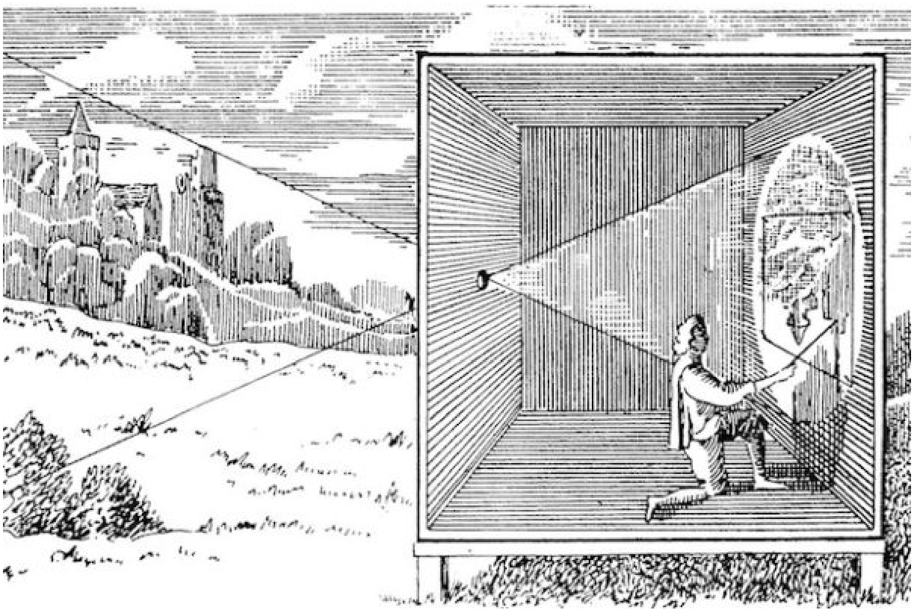


Figure A.1: Old drawings of a "*camera obscura*".

The simplest model used for image acquisition is the pinhole camera (from the Latin "*camera obscura*") model, which is based on the passage of light through a small hole resulting in the projection of the image of an external object onto the opposite plane. The basic working principle was already known back in the time of Mozi (China, 470-390 B.C.) and Aristotle (Greece, 384-322 B.C.) and was used for centuries as drawing support by influential artists such as Leonardo Da Vinci (Italy, 1452-1519 A.C.) (Fig. A.1). The projection of each point is obtained by a straight line passing through a hole, called *optical centre* or *projection centre*, and incident to the *image plane*. Consequently, the projected image appears upside down with respect to the real object, as illustrated in Fig. A.2.

Ideally, each point on the image film where the light is captured is illuminated by a

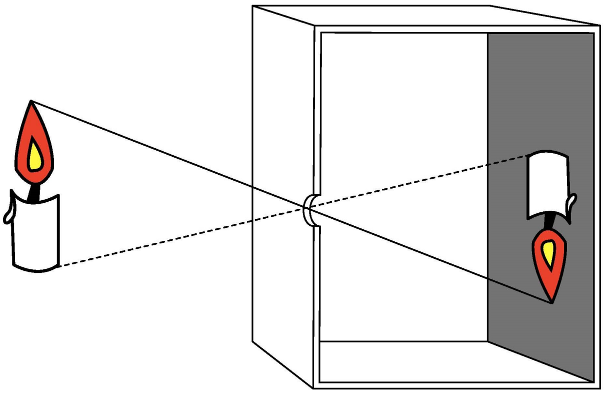


Figure A.2: A graphical example of an ideal pinhole camera.

single ray of light passing by an infinitesimal hole, making the overall picture extremely feeble. In practice, however, the hole has a non-infinitesimal dimension which lets a frustum of light coming from a point in the world pass by, as shown in Fig. A.3. This makes the picture brighter but introduces an undesired blurring effect.

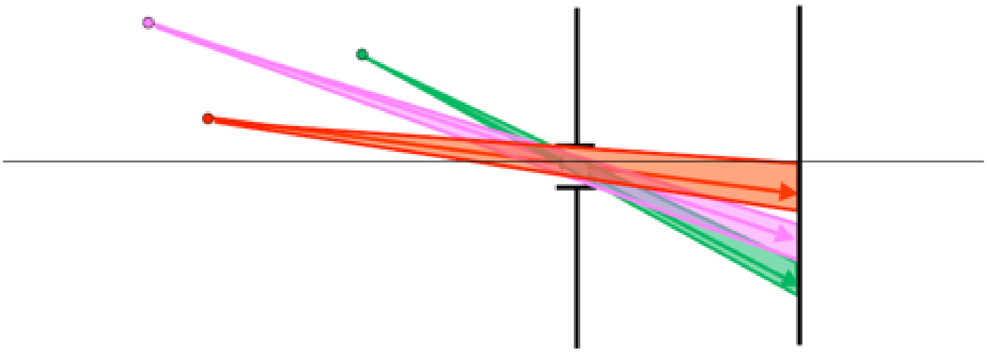


Figure A.3: A graphical example of a realistic pinhole camera.

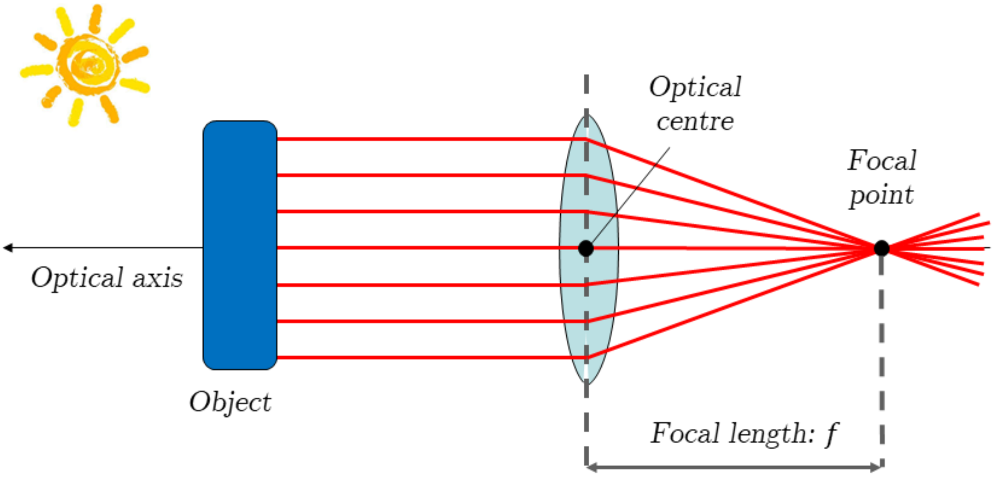


Figure A.4: A graphical example of how a lens suitably deviates the light.

To tackle this issue, a lens is used to focus the light on the film by deviating the frustum coming from each 3D point to a single point on the image. All rays parallel to the *optical axis*, which is the axis perpendicular to the plane of the lens and passing by the optical centre, converge on the so-called *focal point*, as shown in Fig. A.4. The distance between the optical centre and the focal point is called *focal length*.

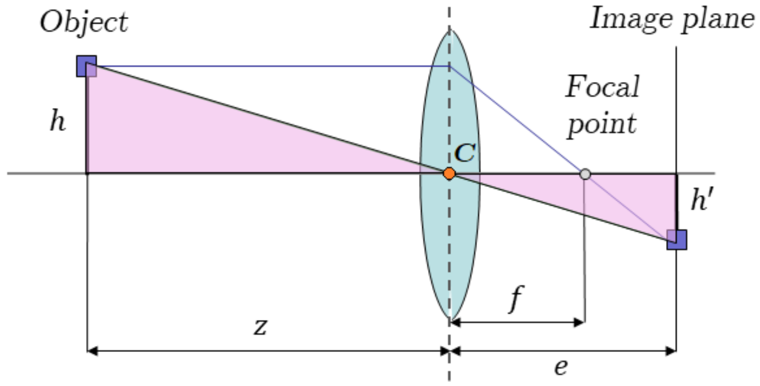


Figure A.5: Similar triangles in pinhole projection.

With reference to Fig. A.5, let us define:

- **C** as the optical centre;
- f as the focal length;
- z as the depth of an object, that is, the distance of the object from the optical centre projected on the optical axis;
- e as the distance from the optical centre to the image plane;
- h as the height of an object in the 3D world;
- h' as the height of the projected object on the image plane.

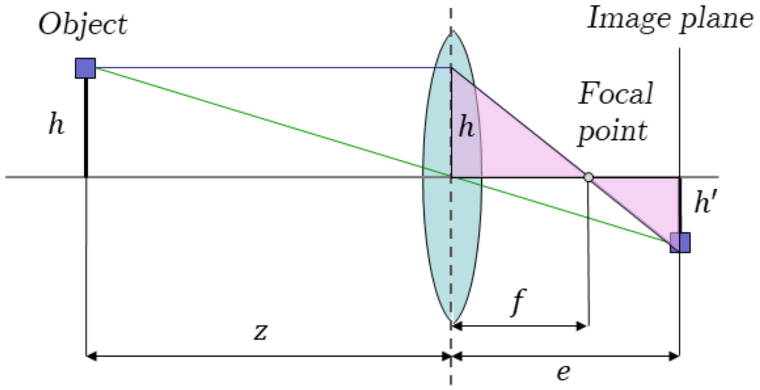


Figure A.6: Similar triangles in pinhole projection.

Considering similar triangles in Fig. A.5 yields:

$$\frac{h'}{h} = \frac{e}{z} \tag{A.1}$$

On the same line, considering similar triangles in Fig. A.6 and Eq. (A.1) yields:

$$\begin{cases} \dfrac{h'}{h} = \dfrac{e}{z} \\ \dfrac{h'}{h} = \dfrac{e-f}{f} = \dfrac{e}{f} - 1 \end{cases} \Rightarrow \frac{e}{f} - 1 = \frac{e}{z} \Rightarrow \frac{1}{f} = \frac{1}{z} + \frac{1}{e} \tag{A.2}$$

which is called the "thin lens" equation. In general, z is much larger than e, so that $1/z$ is negligible compared to $1/e$ and thus:

$$\frac{1}{f} \approx \frac{1}{e} \Rightarrow f \approx e \tag{A.3}$$

The correlation between the real dimension of an object and its depth is called *perspective* and is described by the following relationship:

$$\frac{h'}{h} \approx \frac{f}{z} \Rightarrow h' \approx \frac{f}{z} h \tag{A.4}$$

For convenience, the image plane is often represented in front of the optical centre to preserve the same orientation of the framed scene, that is, upright, as illustrated in Fig. A.7.

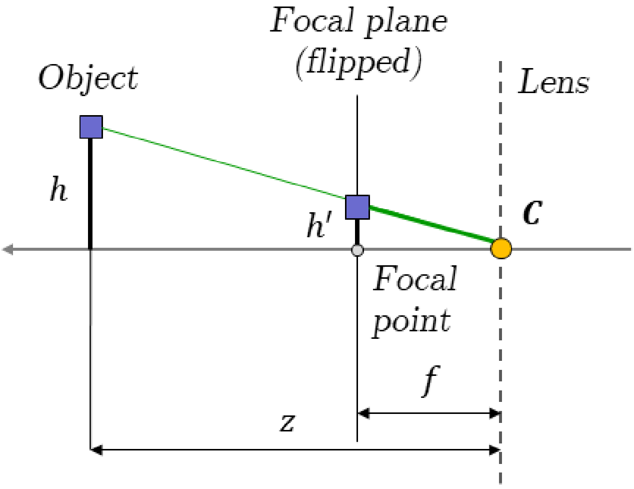


Figure A.7: Convention in pinhole camera projection. Because $f \approx e$, the *focal plane*, which is parallel to the image plane and passing by the focal point, can be considered coincident with the image plane itself.

It must be noticed that a 2D camera, disregarding the model used, does not provide distance measurements but rather the angles with respect to an object. Consequentially, it is considered a bearing sensor, as highlighted by Fig. A.8, where:

- **O** is the principal point obtained by the intersection of the optical axis with the image plane;
- **p** is the point on the image plane resulting from the projection of any point $\mathbf{P}_c, \mathbf{P}'_c, \ldots$ from the same projection line;
- φ is the bearing angle subtended between the optical axis and the projection line of the point **p**; (u, v) are the coordinates system of the image plane, with the origin in the top-left corner of the image as a convention; (x_c, y_c, z_c) are the

coordinates system of the camera, with the origin in **C**; z_c also coincides with the optical axis.

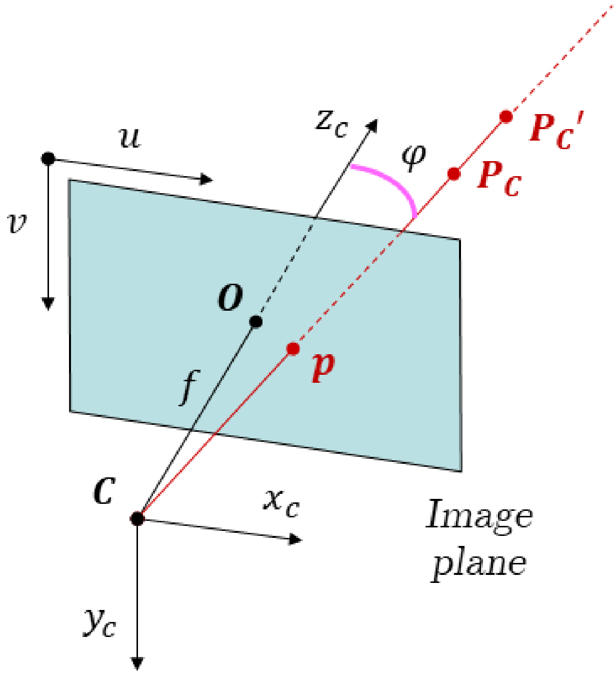


Figure A.8: Pinhole camera as a bearing sensor.

Considering how the model is constructed, there is an inevitable loss of information in the transition from the three-dimensional world to the planar image space concerning the distance that a point $\mathbf{P}_c$ has from the camera. Hence, the inverse transformation from 2D space to 3D space is a problem with infinite solutions. The point in the image plane is, in fact, mapped on a line in space that passes through the projection **p** and the optical centre **C**. The corresponding 3D point $\mathbf{P}_c$ can, therefore, lie anywhere on this line.

On the contrary, the projection of a known 3D point onto the image is a well-posed problem that yields a single solution and whose procedure can be outlined as follows:

1. convert world point $\mathbf{P}_w$, expressed in world coordinates, into the camera coordinates system, $\mathbf{P}_c$;

2. project $\mathbf{P}_c$ onto the image plane and obtain the (x, y) planar coordinates in meters (the planar reference system is oriented like the camera reference system and has its origin in **O**);

3. convert **p** in discrete coordinates (u, v) in the pixel space.

Considering the x_c-coordinate of $\mathbf{P}_c$ and Eq. (A.4), the corresponding x-coordinate in the image plane is obtained as:

$$\frac{x}{f} = \frac{x_c}{z_c} \Rightarrow x = \frac{f x_c}{z_c} \tag{A.5}$$

Similarly, for the y component:

$$\frac{y}{f} = \frac{y_c}{z_c} \Rightarrow y = \frac{f y_c}{z_c} \tag{A.6}$$

To convert $\mathbf{p}$ from the image-plane coordinates (x, y) to the pixel coordinates (u, v), one must consider the pixel coordinates of the principal point $\mathbf{O} = (u_0, v_0)$ and the scale factors k_u, k_v to move from meters to pixels in both dimensions.

$$u = u_0 + k_u x \Rightarrow u = u_0 + \frac{k_u f x_c}{z_c} \tag{A.7}$$

$$v = v_0 + k_v x \Rightarrow v = v_0 + \frac{k_v f y_c}{z_c} \tag{A.8}$$

which can be formulated in matrix form by introducing an extra scale element λ, usually set to 1, to express $\mathbf{p}$ in *homogeneous coordinates* $\tilde{\mathbf{p}}$ and, thus, obtain a linear mapping from 3D to 2D:

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} \longrightarrow \tilde{\mathbf{p}} = \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{A.9}$$

$$\tilde{\mathbf{p}} = \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{P}_c \tag{A.10}$$

or alternatively:

$$\tilde{\mathbf{p}} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{P}_c = \mathbf{K}\mathbf{P}_c \tag{A.11}$$

where:

- α_u is the focal length in u direction in pixels;
- α_v is the focal length in v direction in pixels;
- $\mathbf{K}$ is the *calibration matrix* or, more commonly, the *intrinsic parameters matrix*.

Lastly, if the point is originally expressed in world coordinates $\mathbf{P}_w$, by considering the matrix of extrinsic parameters, which is the roto-translation transformation from world to camera reference systems, one can transform $\mathbf{P}_w$ into camera coordinates as:

$$\mathbf{P}_c = \mathbf{R}\mathbf{P}_w + \mathbf{t} = [\mathbf{R}|\mathbf{t}] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = [\mathbf{R}|\mathbf{t}]\, \tilde{\mathbf{P}}_w \tag{A.12}$$

Putting together Eq. (A.12) and (A.11) yields:

$$\tilde{\mathbf{p}} = \mathbf{K}\, [\mathbf{R}|\mathbf{t}]\, \tilde{\mathbf{P}}_w = \mathbf{M}\tilde{\mathbf{P}}_w \tag{A.13}$$

where $\mathbf{M}$ is called the *projection matrix*. A graphical geometrical representation of the whole process is shown in Fig. A.9.

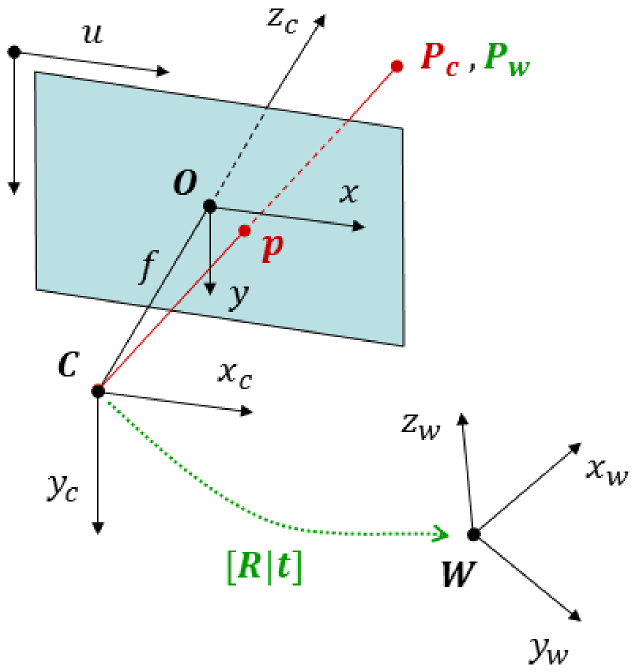


Figure A.9: Graphical geometrical representation of the conversion from 3D world to 2D image.

The presence of a lens usually introduces two types of distortions on the image: a *radial distortion* due to the curvature of the lens, and a *tangential distortion* due to the misalignment between the lens and the image plane. The former one, which is usually the most prominent, can be modelled as a non-linear function of distance from the centre of the optical centre, r. To move from ideal coordinates (u, v) to distorted coordinates (u_d, v_d) a polynomial function if commonly used:

$$u_d = u\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{A.14}$$

$$v_d = v\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{A.15}$$

The following polynomial function is instead commonly used to describe the tangential distortion effect:

$$u_d = u + \left[2p_1 uv + p_2(r^2 + 2u^2)\right] \tag{A.16}$$

$$v_d = v + \left[p_1(r^2 + 2y^2) + 2p_2 uv\right] \tag{A.17}$$

So, in short, there are usually five parameters to estimate the distortion model of a realistic image: $(k_1, k_2, p_1, p_2, k_3)$[1].

The objective of camera calibration, therefore, is to estimate both the intrinsic and the distortion parameters, which together define the projection model of a realistic camera and allow the conversion of a 3D point into the corresponding 2D point in the pixel space. The estimated parameters are specific to each camera with a fixed focal length, and a camera whose parameters are known is commonly said as "calibrated". The calibration is sensitive to the condition of the camera, such as temperature and

[1] The order of the parameters is according to a convention.

wear. Consequentially, it is recommended to re-calibrate a camera sensor at regular intervals. The calibration procedure is based on a *Direct Linear Transformation (DLT)* and requires a calibration pattern (usually a chessboard) as an external aid to be carried out. More detailed information can be found in [77].

Appendix B

Basic notions on ROS

Figure B.1: ROS equation. Source: `https://www.ros.org/blog/ecosystem/`.

Robotic Operating System (ROS) is an open-source Software Development Kit (SDK), which consists in a set of software libraries and tools that help build applications deployable across a wide variety of robotic platforms and contexts. It was originally conceived in 2007 at the *Standford Artificial Intelligence Laboratory* and further developed at *Willow Garage* from 2007 to 2013 when the *Open Source Robotics Foundation (OSRF)* took over and eventually changed the name to *Open Robotics* before becoming the main manager of this world-wide renown ecosystem.

ROS is characterized by four main features, schematized in Fig. B.1:

1. **plumbing**: this is the term commonly used to describe a "middle-ware", i.e. a message-passing system that lets hardware and software components communicate to each other through distributed *nodes* via publish/subscribe patterns; it includes fault isolation, clear interfaces and separation of concerns, and it helps to deploy systems which are easier to maintain and re-utilize;

2. **tools**: developer tools are often crucial to fuel the creation process of a new application; ROS offers launching, introspection, debugging, visualization, plotting, logging and playback functionalities to facilitate this process;

3. **capabilities**: the coverage of ROS libraries is very extensive, from drivers to user interfaces and cutting-edge algorithms; the amount of contributions from all over the globe has made this SDK one of the most complete in the robotics context;

4. **community**: the true power of ROS lies in the global community that continuously orbits around and contributes to it; from students and hobbyists to multinational corporations and government agencies, ROS is entering more and more into robotics-engineering every-day life.

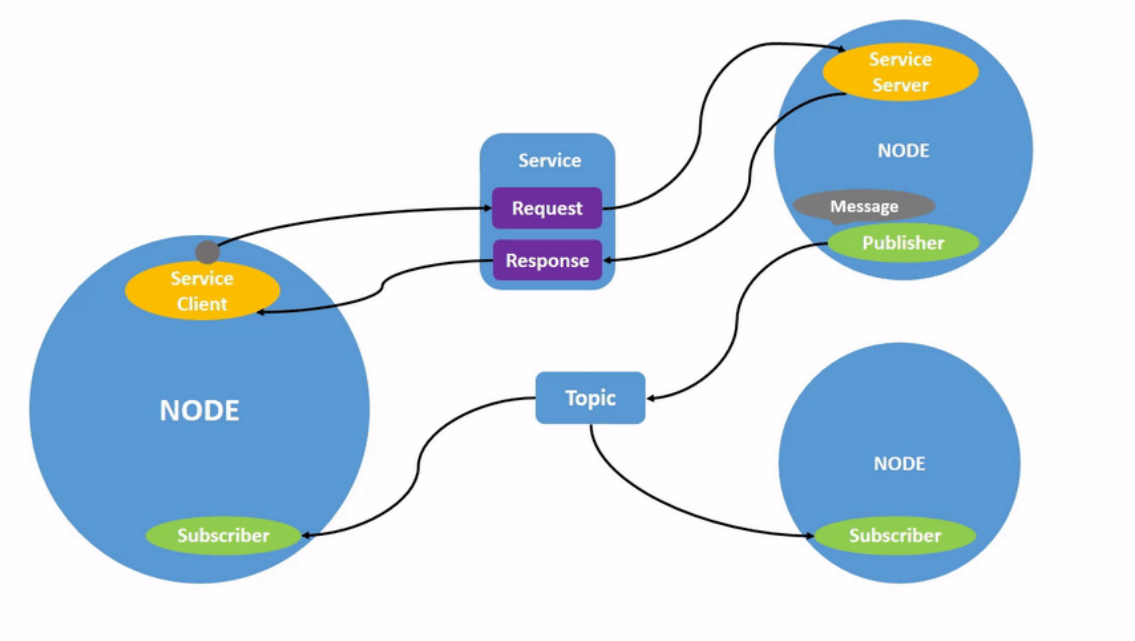


Figure B.2: ROS intra-node basic communication.. Source: `https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html`

ROS is a peer-to-peer system that consists of many small programs (i.e. *node*) which connect to each other and continuously exchange *messages* through *topics* or *services*, as illustrated in Fig. B.2. Each software module can be written in any language for which a client library has been written, such as C++, Python, Java, etc., making ROS a multi-lingual SDK. By ROS convention, contributors are encouraged to create stand-alone libraries/packages and then wrap them so that they can interact with other *ROS modules*.

The core concepts of ROS are:

- **nodes**: single-purposed executable programs, such as an actuator driver, a path planner or a GUI[1], that is individually compiled, executed and managed; they are written using ROS client library available in C++ and Python; *ROS nodes* can *publish* or *subscribe* to a *ROS topic* as well as *provide* or *use* a *ROS service* or a *ROS action*;

- **topics**: named streams of *messages* with a defined type; for instance, data from a camera may be sent on the topic called "`camera_image`" with a *message* of type "`sensor_msgs/Image`"; the publish/subscribe model is a 1-to-N broadcasting system, meaning that a *message* published by a *node* onto a *topic* is received by all *nodes* subscribed to that same *topic*; nevertheless, many *nodes* may publish their own *messages* on the same *topic*, which, in turn, will be received by all subscribers;

- **messages**: strictly-typed data structure for inter-node communication, for instance the *message* of type "`geometry_msgs/Twist`" is a structure including two fields, `linear` and `angular`; each field is itself a *message* of type "`Vector3`" which includes three scalar `float64` values (e.g. `x,y,z`);

[1] Graphical User Interface.

- **services**: synchronous inter-node transactions in the form of blocking Remote Procedure Call (RPC); the service/client model is a 1-to-1 request-response where the client asks for some information and waits for the reply from the server; they are usually used for outsourcing heavy computations, triggering a functionality or behaviour, or retrieving information from other *nodes*;

- **actions**: asynchronous inter-node non-blocking transactions; they are similar to *services*, but they implement the possibility to cancel the request, get periodic feedback during the execution and obtain a final response at task complete; they are particularly useful to execute long-running goals that can be preempted and stacked in a queue;

- **master/core**: it provides a connection information to *ROS nodes* so that they can transmit *messages* to each other; whenever a *node* is activated or launched it immediately connects to a specified *master* to register details of the *message* streams it publishes, *services* and *actions* that it provides, and streams, *services*, an *actions* to which it needs to subscribe; in turn, the *master* provides it with the information needed to form a direct peer-to-peer TCP[2]-based connection with other *nodes* interested by the same *topic* or related to common *services* and/or *actions*.

In jargon, the overall underlying network set up by ROS is called the *ROS cloud*. The software in ROS is organized in *ROS packages*, which contain one or more *ROS nodes*, documentation, configuration files, and provide a ROS interface. A set of *ROS packages* is called a *ROS stack*, and stacks are usually collected in online repositories. The collection of all ROS repositories composes the so-called *ROS-universe*, which, like the real one, is constantly expanding.

[2] Transmission Control Protocol.

Appendix C

TanGent-Angle kernel implementation

In Euclidean geometry, a two-dimensional line in Cartesian coordinates can be described by the well-known algebraic linear equation:

$$y = mx + q \tag{C.1}$$

where:

- m is the *slope* or *gradient* of the line;
- b is the y*-intercept* of the line;
- x is the *independent variable* of the generic function $y = f(x)$.

From trigonometry, the gradient m can also be expressed as the tangent of the counter-clockwise angle α comprised between the line and x-axis of the Cartesian system, as illustrated in Fig. C.1. α is called *tangent* (or *inclination*) *angle*.

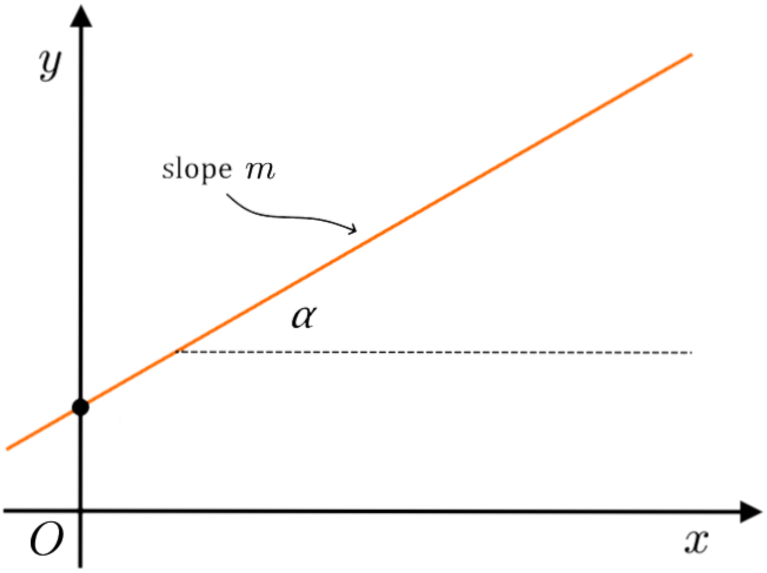
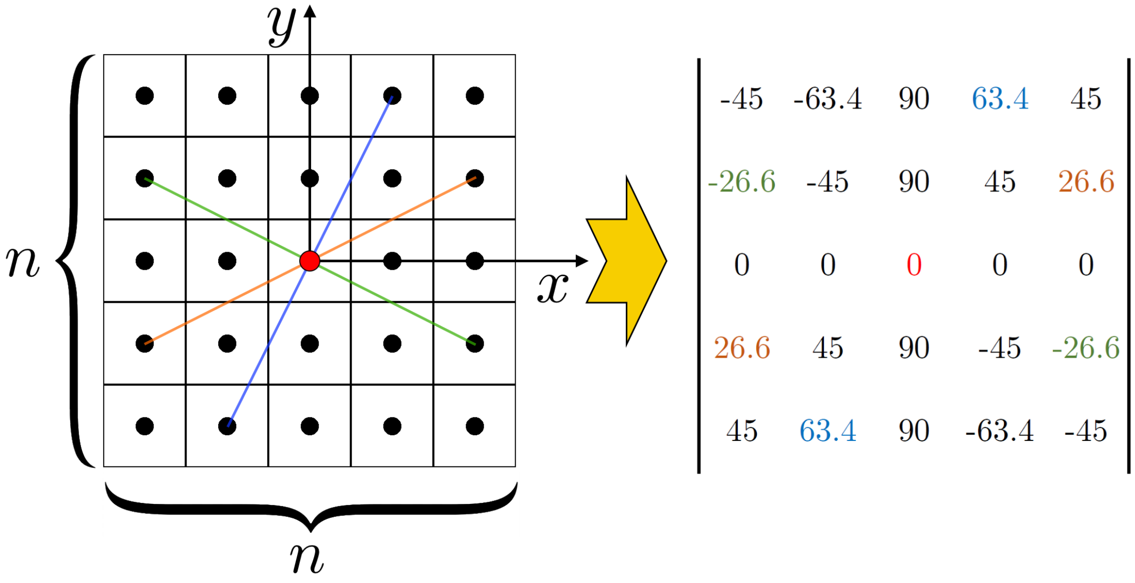


Figure C.1: Inclination angle of a 2D line.

Given two points $(x_1, y_1), (x_2, y_2)$ belonging to the line, α is given by:

$$\tan(\alpha) = m = \frac{y_2 - y_1}{x_2 - x_1} \iff \alpha = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{C.2}$$

Let us now consider a square grid of size $n \times n$ with n, being an odd integer greater than 1, and where each cell has dimension 1×1. By fixing the reference system in the centre of the grid and taking the origin (i.e. the centre cell) as the first point, it is possible to compute the inclination angle for each other cell in the grid by applying Eq. C.1, and build a matrix of tangent angles (in degrees here) of the same dimension as the original grid (see Fig. C.2).



$$\begin{vmatrix} -45 & -63.4 & 90 & 63.4 & 45 \\ -26.6 & -45 & 90 & 45 & 26.6 \\ 0 & 0 & 0 & 0 & 0 \\ 26.6 & 45 & 90 & -45 & -26.6 \\ 45 & 63.4 & 90 & -63.4 & -45 \end{vmatrix}$$

Figure C.2: Example of a TGA kernel of size $n = 5$.

We define the resulting matrix *TanGent Angle (TGA) kernel*. In a more generic algebraic form:

$$\mathbf{K}_{TGA}[i,j] = \begin{cases} 0^\circ & \texttt{,for} \quad i = j = (n-1)/2 \\ \arctan\left(\frac{i - \frac{n-1}{2}}{-j + \frac{n-1}{2}}\right) & \texttt{,otherwise} \end{cases} \tag{C.3}$$

where i is the row index, and j is the column index of an element of the kernel matrix $(i, j \in 0, 1, \ldots, n-1)$.

In a *binary image*, that is an image with only black (= 1) or white (= 0) values, we refer to a *curve* as a set of contiguous black pixels connected by at least one (for curve extremes, Fig. C.3) and at most two (curve inner points, Fig. C.4) adjacent corners or sides.

In Computer Vision, the term *convolution* is often used to describe what is actually a *cross-correlation*, that is defined by the following operation:

$$[\mathbf{K} \star \mathbf{I}](i,j) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \mathbf{K}(u,v)\mathbf{I}(i+u, j+v) \tag{C.4}$$

where $\mathbf{K}$ is a generic kernel of size $n \times n$, $\mathbf{I}$ is a generic one-channel image, and (i, j) identify the pixel position in the image.

As a matter of fact, by convolving the TGA kernel over each point of a binary image curve and dividing by the number of black points in its receptive field[1], one can obtain the average of the tangent angles around that specific point, as illustrated in the example in Fig. C.5.

[1] The region of the input image that the kernel is looking at. For instance, in the purple area in Fig.4.17.

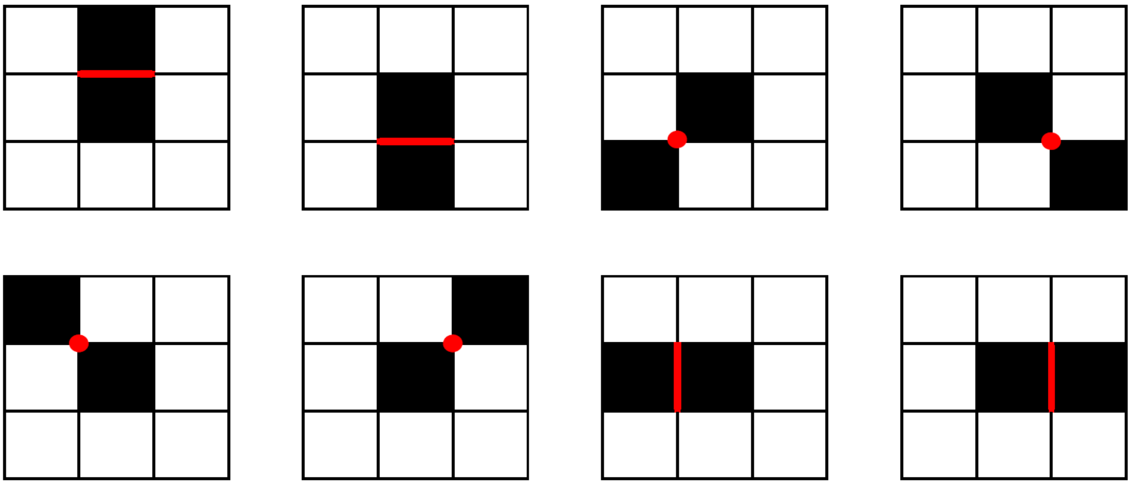


Figure C.3: Topology of possible extremes in a binary digital curve.

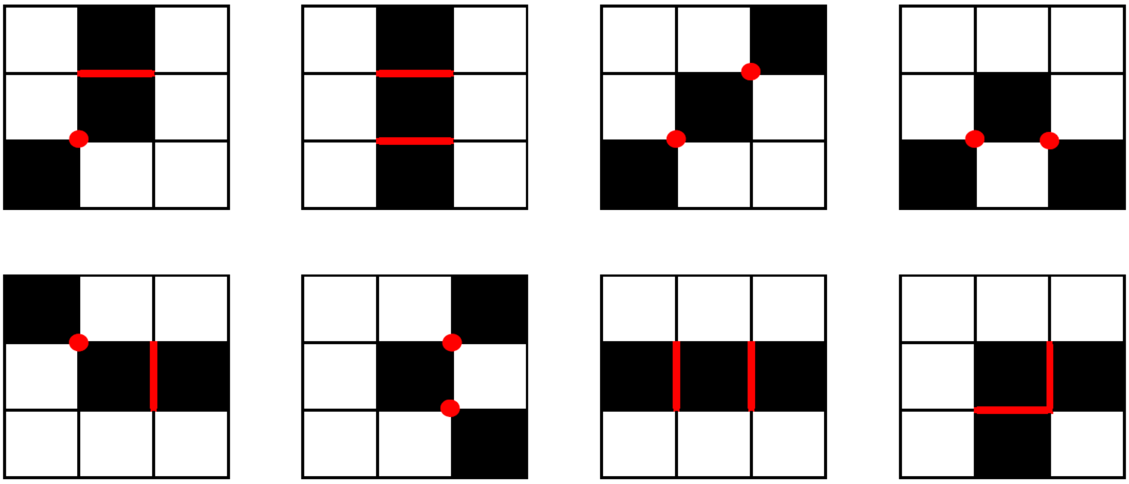


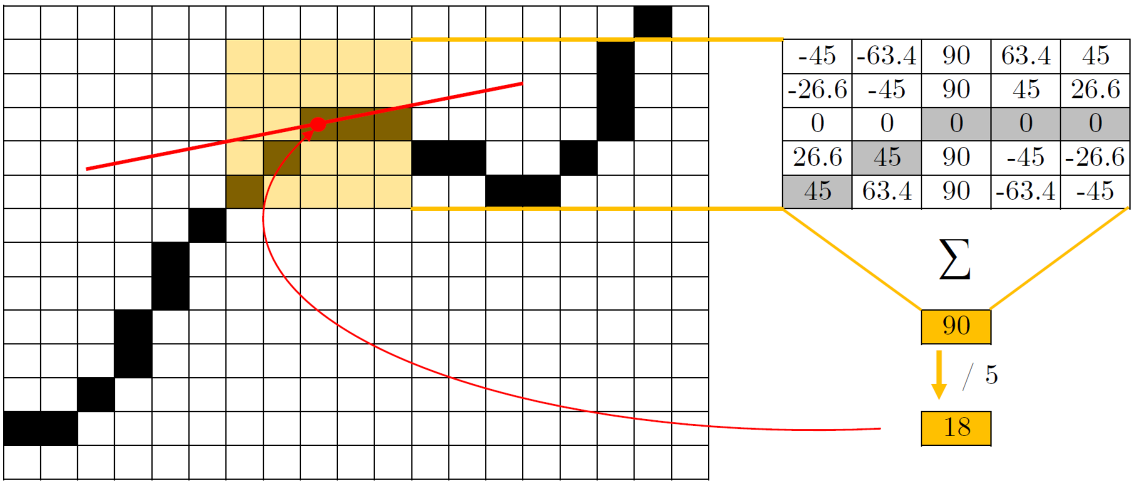
Figure C.4: Topology of possible inner points in a binary digital curve.

A programmatic version of the TGA kernel is illustrated in the Algorithm 1, and consists of four steps:

1. count curve points in the receptive field of each pixel of the binary image;
2. compute the TGA kernel for a given kernel dimension n;
3. apply TGA kernel to the whole image;
4. keep and scale only elements corresponding to curve points in the image.

The level of approximation, that is, the size of the portion of the curve whose tangent one wants to estimate, solely depends on the size of the kernel. Like any smoothing filter, the larger the kernel, the less the noise impact on the resulting value. However, increasing the size too much could lead to a loss of information, such as missing sharp local changes in the curve direction.

Another weakness of this algorithm can be observed in the presence of vertical portions in the curve. In this situation, in fact, the vertical weights of +90° could corrupt the mean value in the case of a descending curve. A possible solution is to split the TGA kernel into two separate kernels of dimension $n \times n$ and $n \times 1$ respectively:



-45	-63.4	90	63.4	45
-26.6	-45	90	45	26.6
0	0	0	0	0
26.6	45	90	-45	-26.6
45	63.4	90	-63.4	-45

Figure C.5: TGA kernel application example.

$$\mathbf{K}_{TGA,1}[i,j] = \begin{cases} 0^\circ & , i = \frac{n-1}{2} j = \frac{n-1}{2} \\ \arctan\left(\frac{i-\frac{n-1}{2}}{-j+\frac{n-1}{2}}\right) & , otherwise \end{cases} \tag{C.5}$$

$$\mathbf{K}_{TGA,2}[i,j] = \begin{cases} 0^\circ & , i = \frac{n-1}{2} \\ 90^\circ & , otherwise \end{cases} \tag{C.6}$$

and use the results of the latter convolution selectively with the changed sign according to the sign of resulting values from the application of the former kernel.

The corrective step appears clearer when looking at the modified version of steps 2 and 3 of the original Algorithm 1 in Algorithm 2, which now becomes:

1. count curve points in the receptive field of each pixel of the binary image;
2. compute the TGA kernels for a given kernel dimension n;

3a. apply the TGA kernels to the whole image;

3b. sum up contributions from both convolutions;

4. keep and scale only elements corresponding to curve points in the image.

Algorithm 1 Image filtering through TGA kernel of size n

```
kernel count ← ones($n, n$)
$C$ ← convolve(input=binary image, kernel=kernel count, mode=’same’, method=’constant’ with 0 paddings)

kernel center ← $n$ div 2
kernel TGA ← zeros($n$, $n$)
for $i$ in range $[0 : n)$ do
    for $j$ in range $[0 : n)$ do
        if $j$ = kernel center and $i$ = kernel center then
            continue
        end if
        kernel TGA $[i, j]$ ← rad2deg(arctan(($i$ – kernel center) / (kernel center - $j$)))
    end for
end for

tangent sums ← convolve(input=binary image, kernel=kernel TGA, mode=’same’, method=’constant’ with 0 paddings)

mask ← where(binary image > 0)
tangent values ← zeroslike(binary image)
tangent values[mask] ← tangent sums[mask] / $C$[mask]
```

Algorithm 2 Image filtering through TGA kernels of size n with vertical line handling

```
kernel center ← $n$ div 2
kernel TGA 1 ← $zeros(n, n)$
kernel TGA 2 ← $zeros(n, 1)$
for $i$ in range [0:$n$) do
    if $i$ ← kernel center then
        continue
    end if
    kernel TGA 1[$i$] ← 90
    for $j$ in range [0 : $n$) do
        if $j$ = kernel center then
            continue
        end if
        kernel TGA 1[$i, j$] ← $rad2deg(arctan((i$ – kernel center) / (kernel center - $j$)))
    end for
end for

tangent sums ← convolve(input=binary image, kernel=kernel TGA 1, mode='same',
method='constant' with 0 paddings)
vertical sums ← convolve(input=binary image, kernel=kernel TGA 2, mode='same',
method='constant' with 0 paddings)

negative mask ← where(tangent sums < 0)
positive vertical sums ← vertical sums
positive vertical sums[negative mask] ← 0
negative vertical sums ← - vertical sums
negative vertical sums[not negative mask] ← 0
tangent sums ← tangent sums + positive vertical sums + negative vertical sums
```

Appendix D

Basic notions on Deep Neural Networks

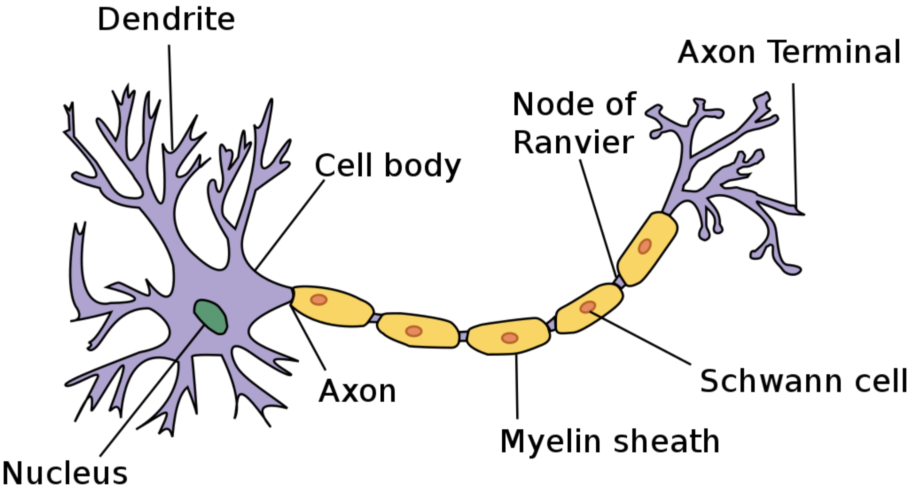
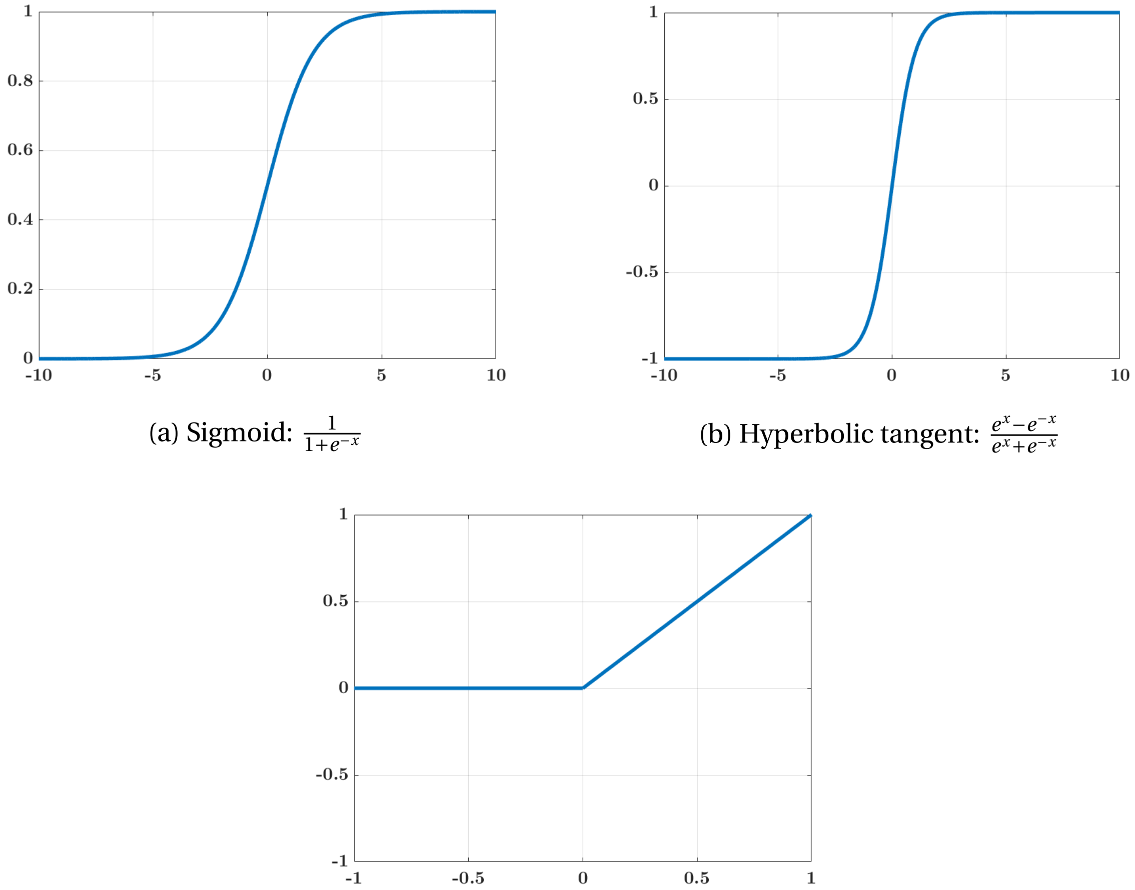


Figure D.1: The cortical neuron. Source: "Anatomy and Physiology" by the US National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program.

A *Neural Network (NN)* is a general-purpose function approximator made up of a series of layers of interconnected units called *(artificial) neurons*, which are inspired by the *cortical neurons*. The biological counterpart, in fact, presents a *dendritic tree* that collects inputs from other neurons, which are summed together, as illustrated in Fig. D.1. When a triggering threshold is exceeded, the *Axon Hillock* generates an electrical impulse that is transmitted through the axon to other neurons. Some cortical neurons are connected to sensorial receptors, and the effect of each input line is controlled by a synaptic weight, which can be positive or negative. The synaptic weights adapt to allow the network, i.e. the brain, to learn how to perform useful actions, and different parts of the cortex are specialized to carry out specific tasks. In a human brain, there are around $2 \cdot 10^{10}$ neurons with nearly 10^{4-5} synapses each. The switching time for a neuron is in the order of 0.001 seconds, which is slow compared to the computing power achieved by recent microchips, and the network is not particularly deep either ($<$ 100 connections). Still, the very high parallelism allows humans to perform extremely complex tasks in a blink of an eye. The understanding of the working principles behind the brain, together with the search for a different paradigm of computation for solving problems difficult to address with traditional algorithmic techniques, are the main motivations that have been fostering the research on NN.

An artificial neuron implements a *logistic regressor* $\sigma(\mathbf{w} \cdot \mathbf{x} + b)$, where $\mathbf{w}$ are the *weights* associated with the inputs $\mathbf{x}$, b is the *bias*, and σ represents an *activation function* that introduces non-linearity and determines the output of the unit. As shown in Fig. D.2, the activation functions normally used are the *sigmoid function*, the *hyperbolic tangent* and the *linear rectifier*, which forms the so-called *Rectified Linear Unit (ReLU)*. The resemblance between the artificial and the biological neuron is evident by looking at Fig. D.3.



(a) Sigmoid: $\frac{1}{1+e^{-x}}$

(b) Hyperbolic tangent: $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

(c) ReLU: if $x > 0$ then x else 0

Figure D.2: Non-linear activation functions.

In addition to the activation function, a NN is also identified by its architecture. In a generic *feed-forward network*, neurons are usually organized in *layers*, and the output of each layer is used as input for the next, as shown, for example, in Fig. D.4. The ends of a network are denoted as the *input* and *output layers*, while the inner ones are called *hidden layers*. If there is more than one hidden layer, the network is *deep* (e.g. Deep Neural Network (DNN)); otherwise, it is called a *shallow* network.

The feed-forward architectures are opposed to the *Recurrent Neural Networks*, where loops, or cycles, are created in which the outputs are carried back as inputs. The number of neurons and the way they are connected together are part of the *hyperparameters*, chosen by the user and fixed before the training phase. Anyways, the interconnection of units in a NN is defined by the parameters of the model, namely weights and biases, that participate in the linear combinations of the inputs to each layer before they are passed to the non-linear activation functions. These parameters are modified

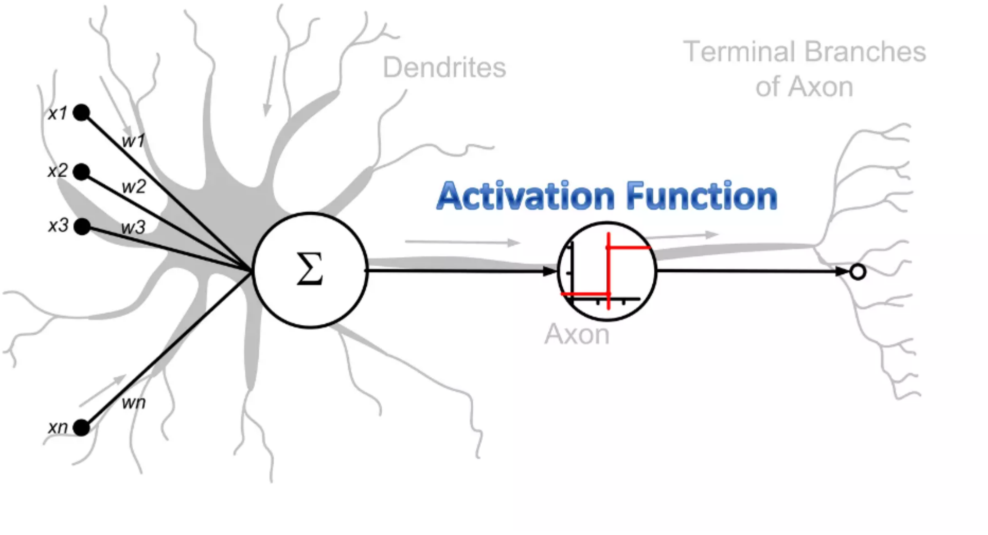


Figure D.3: Resemblance between the artificial and cortical neuron. Credit: Andrew L. Nelson.

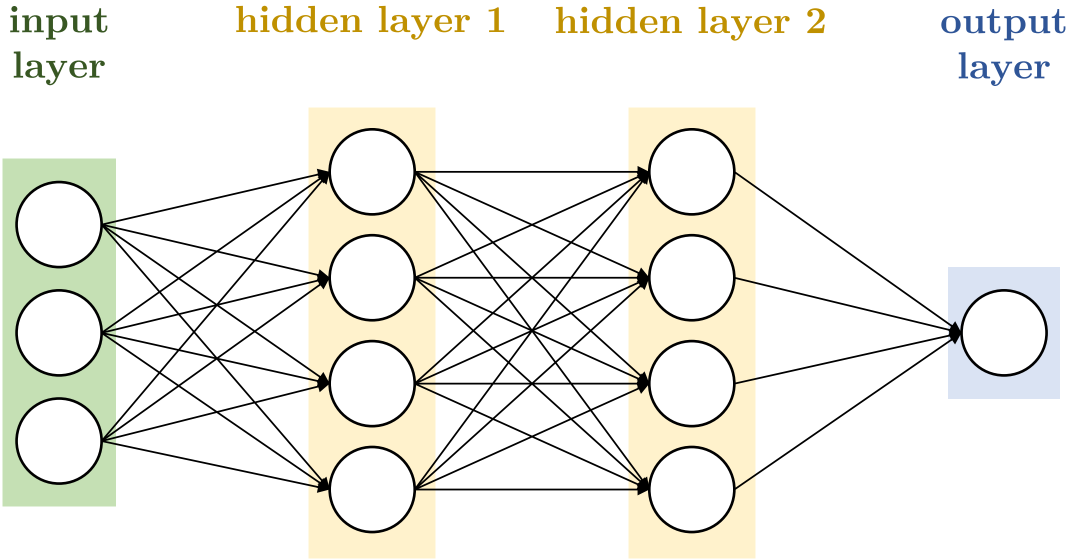


Figure D.4: An example of a feed-forward four-layer DNN.

during the learning phase by being updated in the direction that minimizes a *cost function* (or *loss function*) that expresses the quality of the function approximation given by the NN (for example, the mean squared error). Generally speaking, the main objective of a NN, in fact, is to estimate the best mapping between input-output pairs in the training data. The optimization procedure is called *gradient descent*. The idea behind it is to minimize the loss function by gradually changing the parameters following the opposite direction to the one given by its gradient. According to the notation of [78], b_j^l defines the bias associated with the j-th neuron of layer l, and w_{jk}^l defines the weight given to its input a_k^{l-1}, corresponding to the output of the k-th neuron of layer $l-1$. Considering, for example, sigmoid activations σ, it follows that:

$$a_k^l = \sigma\left(z_j^l\right) = \sigma\left(\sum_k w_{jk} a_k^{l-1} + b_j^l\right) \tag{D.1}$$

Let $\mathscr{L}$ be the loss function and $\Delta\mathscr{L}$ its derivative, then, it holds:

$$\Delta\mathscr{L} \approx \sum_{l,j,k} \frac{\partial \mathscr{L}}{\partial w_{jk}^l} \Delta w_{jk}^l + \frac{\partial \mathscr{L}}{\partial b_j^l} \Delta b_j^l \tag{D.2}$$

or, by vectorizing all parameters in $\boldsymbol{\theta}$:

$$\Delta\mathscr{L} \approx \nabla_{\boldsymbol{\theta}} \mathscr{L} \cdot \Delta\boldsymbol{\theta} \tag{D.3}$$

where $\nabla_{\boldsymbol{\theta}}\mathscr{L}$ is the *total gradient* of the loss function, indicated by $\nabla\mathscr{L}$ for simplicity in the following. Gradient-descent optimization aims at minimizing $\mathscr{L}$ by modifying $\boldsymbol{\theta}$ with:

$$\Delta\boldsymbol{\theta} = -\eta \nabla\mathscr{L} \tag{D.4}$$

where η is the so-called *learning rate*. In this way, $\Delta\mathscr{L} \approx -\eta\|\nabla\mathscr{L}\|^2$, which is never positive.

For practical convenience, *Stochastic Gradient Descent (SGD)* is typically used when training DNNs. A loss function $\mathscr{L}_x$ is, in fact, defined for each input-output pair (x, y), so that $\mathscr{L} = \frac{1}{n}\sum_x \mathscr{L}_x$ and, consequentially, $\nabla\mathscr{L} = \frac{1}{n}\sum_x \nabla\mathscr{L}_x$, with n being the number of training samples in a dataset which is usually very large and, thus, would require a lot of computational effort. Instead of calculating the gradient over all training samples (i.e. *fullbatch*), learning speed can be significantly improved by randomly selecting a subset (i.e. *mini-batch*) of $m < n$ random training samples $x_1, \ldots, x_m$, and approximating $\nabla\mathscr{L}$ by averaging only their contribution:

$$\nabla\mathscr{L} = \frac{1}{n}\sum_{i=1}^{n} \nabla\mathscr{L}_{x_i} \approx \frac{1}{m}\sum_{x=1}^{m} \nabla\mathscr{L}_{x_i} \tag{D.5}$$

Finally, the parameters of the NN are updated as follows:

$$w_{jk}^l \rightarrow {w'}_{jk}^{l} = w_{jk}^l - \frac{\eta}{m}\sum_{i=1}^{m} \frac{\partial \mathscr{L}_{x_i}}{\partial w_{jk}^l} \tag{D.6}$$

$$b_j^l \rightarrow {b'}_j^{l} = b_j^l - \frac{\eta}{m}\sum_{i=1}^{m} \frac{\partial \mathscr{L}_{x_i}}{\partial b_j^l} \tag{D.7}$$

by sweeping randomly chosen mini-batches across multiple *epochs* of training, where an epoch corresponds to one training cycle using all available training data exactly once. That means that each mini-batch is passed through a single pass consisting of a forward pass and a backward pass before each parameters update step.

Larger batches provide smoother estimations and better exploit parallel hardware even though memory requirements scale linearly with the batch size. Usually, power-of-2 sizes are more suitable for parallel hardware, and typical values on single GPUs[1] ranges from 16 to 256, even if modern distributed training can use values up to 8192

[1] Graphics Processing Unit.

or 16384 on multiple GPUs [79]. On the other hand, smaller batches may have a regularization[2] effect and result in better generalization at the expense of higher run time [80].

The computation of the gradient of the loss function is carried out through the so-called *back-propagation* algorithm. As the name suggests, after a feed-forward computation of layers activations, it back-propagates across all the layers the information about how much the cost function is willing to change as a result of a modification in the network. Intuitively, the reason behind the algorithm is that modifying a weight inevitably influences all the activations of neurons which follow that connection, directly or indirectly. Back-propagation traces the parameter responsible for a change in the loss function by starting from output activations and going up this chain of dependence. To prevent the network from overfitting the training data as well as the exploding gradient problem, the weights are progressively shrunk by adding a penalty term to the loss function. This regularization technique is called *weight decay*, and the most common variants are implemented with the L2-norm (Eq. (D.8)) and with the L1-norm (Eq. (D.9)), in order or popularity. By defining with α the *regularization rate* (or weight decay), which is a new hyper-parameter, the loss function respectively becomes:

$$\hat{\mathscr{L}}_{\mathrm{L2}} = \frac{\alpha}{2}\|\mathbf{w}\|_2^2 + \mathscr{L} \tag{D.8}$$

$$\hat{\mathscr{L}}_{\mathrm{L1}} = \alpha\|\mathbf{w}\|_1 + \mathscr{L} \tag{D.9}$$

Despite its advantages, the SGD method is subject to oscillations during the learning phase when using a high learning rate. To counteract this drawback, a lower learning rate could be used. Still, on top of the slower training time, this strategy may lead the optimization to be noisy and/or attracted to critical points such as saddle points or local minima. *Momentum* is one of the most used modifications to SGD to account for this [81]. By adding a "velocity" term $\boldsymbol{v}$, the update rule becomes:

$$\boldsymbol{v}^{t+1} = \beta\boldsymbol{v}^t - \eta\nabla\mathscr{L}(\boldsymbol{\theta}^t) \tag{D.10}$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \boldsymbol{v}^{t+1} \tag{D.11}$$

Intuitively, SGD moves like a rolling ball down a surface that acquires speed (e.g. "momentum") in the downward direction and does not immediately loses velocity when the surface changes curvature. An important variant of momentum is the *Nesterov Accelerated Gradients*, proposed in 1983 [82]. It is very similar to the momentum, but the gradient is computed after having "partially" updated $\boldsymbol{\theta}^t$ with $\beta\boldsymbol{v}^t$. In simple words, it "looks ahead" to compute a more accurate gradient so that Eq. (D.10) becomes:

$$\boldsymbol{v}^{t+1} = \beta\boldsymbol{v}^t - \eta\nabla\mathscr{L}(\boldsymbol{\theta}^t + \beta\boldsymbol{v}^t) \tag{D.12}$$

A different approach to improve SGD is to define per-parameter adaptive learning rates. This method tends to work reasonably well even when its hyper-parameters are not perfectly tuned, differently from the momentum, which, when properly tuned, typically leads to better solutions. *Adaptive Gradient (AdaGrad)* [83] proposes to rescale

[2] *Regularization* is a set of techniques that can prevent *overfitting* in NNs and, thus, improve the accuracy of a Deep Learning model when facing completely new data from the problem domain.

each entry of the gradient with the inverse of the history of its squared values. Consequentially, weights receiving small gradients will have their effective learning rate increased; vice-versa, weights receiving large gradients will have their effective learning rate decreased. The main drawback of AdaGrad is that it may reduce all learning rates too early when the optimization is still far from a good minimum. *ADAptive Moments (ADAM)* [84] propose a method to solve this issue by down-weighing the history of the past and keeping the optimizer responsive by introducing a running average of the scaling factor and the gradients, which acts like momentum.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] T. Mazali and A. Magone, *Industria 4.0 - Uomini e macchine nella fabbrica digitale*, Guerini e Associati, Ed. Jan. 2016, ISBN: 978-88-6250-638-0.

[2] R. Galin and R. Meshcheryakov, "Automation and robotics in the context of Industry 4.0: the shift to collaborative robots," in *IOP Conference Series: Materials Science and Engineering*, vol. 537, Krasnoyarsk, RU: IOP Publishing, May 2019, p. 032 073.

[3] M. Bortolini, E. Ferrari, M. Gamberi, F. Pilati, and M. Faccio, "Assembly system design in the Industry 4.0 era: a general framework," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5700–5705, Jul. 2017, ISSN: 2405-8963.

[4] C. Hinojosa and X. Potau, "Advanced industrial robotics: Taking human-robot collaboration to the next level," European Foundation for the Improvement of Living and Working Conditions, Tech. Rep. Jan. 2018.

[5] N. Magnavita, "Productive aging, work engagement and participation of older workers. a triadic approach to health and safety in the workplace.," *Epidemiology, Biostatistics and Public Health*, vol. 14, 2 Jul. 2017.

[6] "Robots and robotic devices — Vocabulary," International Organization for Standardization (ISO), Tech. Rep. 8373, 2012.

[7] N. Zonato, "Interazione uomo macchina e robot collaborativi: Sviluppi recenti e problematiche," Master Thesis, Università degli studi di Padova, 2015.

[8] "Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design," International Organization for Standardization (ISO), Tech. Rep. 13849-1, 2015.

[9] "Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots," International Organization for Standardization (ISO), Tech. Rep. 10218-1, 2011.

[10] "Robots and robotic devices — Safety requirements for industrial robots — Part 2: Robot systems and integration," International Organization for Standardization (ISO), Tech. Rep. 10218-2, 2011.

[11] "Robots and robotic devices — Collaborative robots," International Organization for Standardization (ISO), Tech. Rep. 15066, 2016, Technical Specification (TS).

[12] N. Pedrocchi, F. Vicentini, M. Matteo, and L. M. Tosatti, "Safe human-robot cooperation in an industrial environment," *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, p. 27, 2013.

[13] K. Berntorp, K.-E. Arzen, and A. Robertsson, "Mobile manipulation with a kinematically redundant manipulator for a pick-and-place scenario," in *2012 IEEE International Conference on Control Applications (CCTA)*, Dubrovnik, HR: IEEE, Oct. 2012, pp. 1596–1602, ISBN: 978-1-4673-4503-3.

[14] M. Nieuwenhuisen, D. Droeschel, D. Holz, *et al.*, "Mobile bin picking with an anthropomorphic service robot," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, DE: IEEE, May 2013, pp. 2327–2334.

[15] G. Michalos, S. Makris, J. Spiliotopoulos, I. Misios, P. Tsarouchi, and G. Chryssolouris, "ROBO-PARTNER: Seamless human-robot cooperation for intelligent, flexible and safe operations in the assembly factories of the future," *Procedia CIRP*, vol. 23, pp. 71–76, 2014, 5th CATS 2014 - CIRP Conference on Assembly Technologies and Systems, Patras, EL, ISSN: 2212-8271.

[16] R. Krug, T. Stoyanov, V. Tincani, *et al.*, "The next step in robot commissioning: Autonomous picking and palletizing," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 546–553, Jan. 2016.

[17] J. Saenz, F. Penzlin, C. Vogel, and M. Fritzsche, "VALERI - A Collaborative Mobile Manipulator for Aerospace Production," in *Advances in Cooperative Robotics: Proceedings of the 19th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, London, UK: World Scientific, 2016, pp. 186–195.

[18] A. Dömel, S. Kriegel, M. Kassecker, M. Brucker, T. Bodenmüller, and M. Suppa, "Toward fully autonomous mobile manipulation for industrial environments," *International Journal of Advanced Robotic Systems*, vol. 14, Jul. 2017.

[19] V. V. Unhelkar, S. Dörr, A. Bubeck, *et al.*, "Mobile Robots for Moving-Floor Assembly Lines: Design, Evaluation, and Deployment," *IEEE Robotics Automation Magazine*, vol. 25, no. 2, pp. 72–81, Jun. 2018, ISSN: 1558-223X.

[20] F. Zaccaria, A. Baldassarri, G. Palli, and M. Carricato, "A Mobile Robotized System for Depalletizing Applications: Design and Experimentation," *IEEE Access*, vol. 9, pp. 96 682–96 691, 2021.

[21] J. Aleotti, A. Baldassarri, M. Bonfè, *et al.*, "Toward Future Automatic Warehouses: An Autonomous Depalletizing System Based on Mobile Manipulation and 3D Perception," *Applied Sciences*, vol. 11, no. 13, 2021, ISSN: 2076-3417.

[22] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning," *Applied Sciences*, vol. 9, no. 2, p. 348, Jan. 2019, ISSN: 2076-3417.

[23] M. Yang, E. Yang, R. C. Zante, M. Post, and X. Liu, "Collaborative mobile industrial manipulator: A review of system architecture and applications," in *25th International Conference on Automation and Computing (ICAC)*, Lancaster, UK: IEEE, Sep. 2019, pp. 1–6.

[24] Z. M. Bi, C. Luo, Z. Miao, B. Zhang, W. J. Zhang, and L. Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102 022, 2021, ISSN: 0736-5845.

[25] (. R. Davies E. R., *Computer and machine vision: theory, algorithms, practicalities*, eng, Fourth. Waltham, Mass.: Elsevier, 2012, ISBN: 0-12-809575-X.

[26] A. N. Belbachir, *Smart Cameras*, First, A. N. Belbachir, Ed. New York, NY, USA: Springer, Oct. 2009, ISBN: 978-1-4419-0953-4.

[27] E. R. Fossum and D. B. Hondongwa, "Review of the Pinned Photodiode for CCD and CMOS Image Sensors," *IEEE Journal of the Electron Devices Society*, vol. 2, no. 3, pp. 33–43, 2014.

[28] S. Meroli, *CMOS vs CCD sensor. Who is the clear winner?* `https://meroli.web.cern.ch/lecture_cmos_vs_ccd_pixel_sensor.html`, Accessed: 2022-12-28.

[29] A. Mertan, D. Duff, and G. Unal, "Single Image Depth Estimation: An Overview," *Digital Signal Processing*, vol. 123, p. 103 441, Jan. 2022.

[30] K. Tateno, F. Tombari, and N. Navab, "When 2.5D is Not Enough: Simultaneous Reconstruction, Segmentation and Recognition on Dense SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, SE: IEEE, 2016, pp. 2295–2302.

[31] S. Giancola, M. Valenti, and R. Sala, "Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies," in *SpringerBriefs in Computer Science*, ser. 2191-5768, Cham, CH: Springer International Publishing, 2018.

[32] J. Walsh, N. O' Mahony, S. Campbell, *et al.*, "Deep Learning vs. Traditional Computer Vision," in *Computer Vision Conference (CVC) 2019*, Las Vegas, NV, USA, Apr. 2019, ISBN: 978-981-13-6209-5.

[33] J. Tanz, *Soon We Won't Program Computers. We'll Train Them Like Dogs*, WIRED, Accessed on 2022-12-29, May 2017.

[34] E. Pedrosa, G. H. Lim, F. Amaral, *et al.*, "TIMAIRIS: Autonomous Blank Feeding for Packaging Machines," in *Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users: The Experience of the European Robotics Challenges*, F. Caccavale, C. Ott, B. Winkler, and Z. Taylor, Eds. Cham, CH: Springer International Publishing, 2020, pp. 153–186, ISBN: 978-3-030-34507-5.

[35] S. Comari, R. Di Leva, M. Carricato, *et al.*, "Mobile cobots for autonomous raw-material feeding of automatic packaging machines," *Journal of Manufacturing Systems*, vol. 64, pp. 211–224, 2022, ISSN: 0278-6125.

[36] W. Zhang and C. van Luttervelt, "Toward a resilient manufacturing system," *CIRP Annals*, vol. 60, no. 1, pp. 469–472, 2011, ISSN: 0007-8506.

[37] G. Darragjati, "Studio di fattibilità di macchina per il lavaggio di contenitori farmaceutici," Master Thesis, University of Bologna, Department of Mechanical Engineering, 2011.

[38] D. Coleman, I. A. Şucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3–16, May 2014.

[39] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012, `https://ompl.kavrakilab.org`.

[40] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, 2012, pp. 3859–3866.

[41] J. J. Kuffner and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, San Francisco, CA, USA: IEEE, 2000, pp. 995–1001.

[42] A. Pupa, W. Van Dijk, and C. Secchi, “A Human-Centered Dynamic Scheduling Architecture for Collaborative Application,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4736–4743, 2021.

[43] Q. Zhou, J. Park, and V. Koltun, “Open3D: A Modern Library for 3D Data Processing,” Tech. Rep. 2018, `http://www.open3d.org/`.

[44] Q. Zhou, *PyMesh - Geometry Processing Library for Python*, `https://pymesh.readthedocs.io/en/latest/`, 2018.

[45] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[46] A. Paszke, S. Gross, F. Massa, *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., 2019.

[47] A. Pupa, M. Arrfou, G. Andreoni, *et al.*, “ROSSINI: a new paradigmfor human-robot interaction.”

[48] S. Comari and M. Carricato, “Vision-based robotic grasping of reels for automatic packaging machines,” *Applied Sciences*, vol. 12, no. 15, 2022, ISSN: 2076-3417.

[49] R. Y. Tsai and R. K. Lenz, “A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, Jun. 1989, ISSN: 2374-958X.

[50] Haiyuan Wu, Qian Chen, and T. Wada, “Conic-based algorithm for visual line estimation from one image,” in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, Seoul, KR, May 2004, pp. 260–265.

[51] R. Brown, 2020.

[52] W. Gander, G. H. Golub, and R. Strebel, “Least-squares fitting of circles and ellipses,” *BIT Numerical Mathematics*, vol. 34, no. 4, pp. 558–578, 1994, ISSN: 1572-9125.

[53] Y. Xie and Q. Ji, “A new efficient ellipse detection method,” in *Object recognition supported by user interaction for service robots*, vol. 2, Aug. 2002, pp. 957–960.

[54] C. A. Basca, M. Talos, and R. Brad, “Randomized Hough Transform for Ellipse Detection with Result Clustering,” in *EUROCON 2005 - The International Conference on "Computer as a Tool"*, vol. 2, Belgrade, CS, Nov. 2005, pp. 1397–1400.

[55] S. Comari and M. Carricato, “Autonomous scanning and cleanliness classificationof the surface of a pharmaceutical bin through AI and robotics.”

[56] L. Li, R. Wang, and X. Zhang, “A Tutorial Review on Point Cloud Registrations: Principle, Classification, Comparison, and Technology Challenges,” English, *Mathematical Problems in Engineering*, vol. 2021, no. 9953910, 2021.

[57] F. Pomerleau, F. Colas, and R. Siegwart, *A Review of Point Cloud Registration Algorithms for Mobile Robotics*. Now Foundations and Trends, 2015, vol. 4.

[58] G. K. L. Tam, Z.-Q. Cheng, Y.-K. Lai, *et al.*, "Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, pp. 1199–1217, 2013.

[59] L. R. de Castro, "6 DoF tool path generator from CAD model for visual inspection of part surfaces," Master Thesis, University of Porto, Engineering faculty, 2022.

[60] G. H. Tarbox and S. N. Gottschlich, "Planning for Complete Sensor Coverage in Inspection," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 84–111, Jan. 1995, ISSN: 1077-3142.

[61] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. The MIT Press, Mar. 2019, ISBN: 9780262039420.

[62] A. Kundu, Y. Li, and J. M. Rehg, "3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Juan, PR, USA: IEEE, Jun. 2018.

[63] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970, ISSN: 0006-3444.

[64] D. L. Applegate, R. E. Bixby, V. Chvatál, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study* (Princeton Series in Applied Mathematics). Princeton University Press, 2006, ISBN: 9780691129938.

[65] C. Dahiya and S. Sangwan, "Literature review on travelling salesman problem," *International Journal of Research*, vol. 5, no. 16, pp. 1152–1155, 2018.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[67] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision - 13th European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science (LNCS, volume 8692), Zurich, CH: Springer International Publishing, 2014, pp. 818–833, ISBN: 978-3-319-10590-1.

[68] U. Michelucci, "Fundamentals of Convolutional Neural Networks," in *Advanced Applied Deep Learning : Convolutional Neural Networks and Object Detection*. Berkeley, CA, USA: Apress, 2019, ch. 3, pp. 79–123, ISBN: 978-1-4842-4976-5.

[69] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *International Conference on Engineering and Technology (ICET)*, Antalya, TR, 2017, pp. 1–6.

[70] C. Liu, R. Chen, K. Chen, and J. Xu, "Ellipse detection using the edges extracted by deep learning," *Machine Vision and Applications*, vol. 33, no. 4, p. 63, 2022, ISSN: 1432-1769.

[71] D.-Y. Kang, P. Duong, and J.-C. Park, "Application of Deep Learning in Dentistry and Implantology," *The Korean Academy of Oral and Maxillofacial Implantology*, vol. 24, pp. 148–181, Sep. 2020.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012, pp. 1106–1114.

[73] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Computational and Biological Learning Society, 2015, pp. 1–14.

[74] C. Szegedy, W. Liu, Y. Jia, *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1–9.

[75] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[76] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, “A survey of robot manipulation in contact,” *Robotics and Autonomous Systems*, vol. 156, Oct. 2022, ISSN: 0921-8890.

[77] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[78] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[79] P. Goyal, P. Dollár, R. B. Girshick, *et al.*, “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour,” Tech. Rep. 2017.

[80] S. L. Smith, E. Elsen, and S. De, “On the Generalization Benefit of Noise in Stochastic Gradient Descent,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML’20, JMLR.org, 2020.

[81] B. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964, ISSN: 0041-5553.

[82] Y. Nesterov, “A method of solving a convex programming problem with convergence rate $O(1/k^2)$,” in *Soviet Mathematics Doklady*, vol. 27, 1983, pp. 372–376.

[83] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011.

[84] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, May 2015.